

EXPLOITING ASYNCHRONY IN GPS RECEIVER SYSTEMS TO ENABLE
ULTRA-LOW-POWER OPERATION

A Dissertation

Presented to the Faculty of the Graduate School
of Cornell University

In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by

Benjamin Zhong Xian Tang

January 2014

© 2014 Benjamin Zhong Xian Tang

EXPLOITING ASYNCHRONY IN GPS RECEIVER SYSTEMS TO ENABLE ULTRA LOW POWER OPERATION

Benjamin Zhong Xian Tang, Ph. D.

Cornell University 2014

Many complex systems are inherently asynchronous. One simple example is the neural network in human bodies where neurons communicate with one another through synaptic signaling. They pass messages around efficiently without any dictation from a global timing reference. Another example is the GPS receiver system; though the satellite constellation is synchronized by atomic clocks, asynchrony is inherent in the operation of the receivers. Forcing some form of clocking to an asynchronous system will not only render it unnatural, but also makes it wasteful of resources. The same can be said about clocking an inherently asynchronous system using conventional VLSI techniques. This dissertation presents the concept and implementation of an asynchronous GPS baseband processor architecture designed for low power applications. All subsystems run at their natural frequencies without clocking and all signal processing is done on-the-fly. Transistor-level simulations and measured power from system components in a 90nm process show that the system only consumes about 1.4mW during continuous tracking. The system also has 3-D rms error below 4 meters, comparing favorably to other contemporary GPS baseband processors.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Prof. Sunil Bhawe, for his guidance, support and encouragement throughout the course of my doctoral degree. He constantly engages me in discussions on cutting-edge technologies in the areas of GPS, MEMS, and Physics and encourages me to explore them to discover new applications and to push technological boundaries not only in low-power navigation, but also in VLSI, encryption, authentication and quantum computing. I am also grateful that he encourages me to pursue the very best to realize my potential while also genuinely looks out for my well-being and development as a person and as a professional.

I would also like to thank my thesis co-advisor, Prof. Rajit Manohar, for his advice and support. His expertise in asynchronous VLSI has provided guidance in my research on the low power asynchronous GPS baseband processor. His CAD tools have also enabled me to fully implement the chip from design inception all the way to tapeout. It has been delightful to learn from his breadth and depth of knowledge in asynchronous VLSI, conventional VLSI and circuit tricks or topologies for advanced energy-efficient and high-performance circuits.

I am also thankful to another member of my graduate committee, Prof. Mark Psiaki, for his advice on GPS algorithms. His insightful questions challenge me to think through my algorithms and his teachings on Estimation and Kalman Filtering allow me to understand and discover more sophisticated algorithms for GPS and sensor integration.

I am also extremely grateful to the late Prof. Paul Kintner Jr. for his vision and advice on GPS systems. His collaborative effort with Prof. Sunil Bhawe and Prof. Rajit Manohar started me off with the low power GPS receiver initiative.

Unfortunately, he passed away too early to witness the product of his vision, the fruition of my research work and any future development henceforth.

In addition, I would like to thank Stephen Longfield Jr. for working with me on certain parts of the asynchronous GPS baseband processor. I am also thankful for the friendship, help and support of the members of the OxideMEMS group, Asynchronous VLSI (AVLSI) group and the Cornell GPS Laboratory (GPSL).

I would also like to thank my parents and my brothers for their love and support. Last but not least, I thank God for His grace that has seen me through the ups and downs in life.

TABLE OF CONTENTS

CHAPTER 1 – INTRODUCTION	12
1.1 MOTIVATION	12
1.2 RELATED WORK	14
CHAPTER 2 – GPS SIGNAL STRUCTURE.....	16
2.1 GPS L1 C/A.....	16
2.2 ASYNCHRONY IN BASEBAND PROCESSING	18
2.3 ASYNCHRONOUS VLSI.....	20
2.3.1 BUNDLED-DATA	21
2.3.2 QDI.....	22
CHAPTER 3 – SYSTEM OVERVIEW	24
3.1 ACQUISITION	25
3.2 TRACKING.....	26
3.3 DATA EXTRACTION	28
CHAPTER 4 – OPTIMIZATIONS	30
4.1 NATURAL FREQUENCY OPERATION OF ALL SUB-SYSTEMS.....	30
4.2 ADDER AND INCREMENT-DECREMENT ACCUMULATORS	30
4.3 MEMORYLESS CODE PHASE ACQUISITION	34
4.4 BUNDLED-DATA MATH.....	38
4.5 OPTIMIZED TRACKING LOOPS.....	39
CHAPTER 5 – CHIP IMPLEMENTATION	44
5.1 STANDARD AND NON-STANDARD CELL FLOW.....	44
5.2 METAL LAYER PLANNING	50
5.3 BUFFERING	50
5.4 POWER-GRID REINFORCEMENT	51
CHAPTER 6 – RESULTS	56
6.1 RECEIVER PERFORMANCE SIMULATIONS.....	56
6.2 POWER SIMULATIONS.....	59
6.3 POST-SILICON TESTING AND MEASUREMENTS	62
6.3.1 TEST STRUCTURES.....	64

6.3.2 FULL SYSTEM TEST	68
CHAPTER 7 – FUTURE	73
7.1 CMOS TECHNOLOGY SCALING	73
7.2 MEMS/NEMS	77
7.2.1 NEM RELAY BASICS	77
7.2.2 CASE FOR ASYNCHRONOUS NEMS	79
7.2.3 QDI SYNTHESIS WITH NEM RELAYS	84
7.2.4 QDI TEMPLATES USING NEM RELAYS	89
7.3 ADVANCED CORRELATORS	96
7.3.1 NEMS CORRELATORS	96
7.3.2 WEAK-SIGNAL INTEGRATION	96
7.3.3 GNSS INTEROPERABILITY	97
CONCLUSIONS.....	98
BIBLIOGRAPHY	100

LIST OF FIGURES

Figure 1 – GPS L1 C/A signal structure.....	16
Figure 2 – Baseband processing's attempt to replicate the satellite's atomic clock time through the GPS signal structure.....	18
Figure 3 – Two common asynchronous handshake protocols: (a) Dual-rail QDI (b) Bundled-data.....	21
Figure 4 – System diagram.....	24
Figure 5 – Increment-decrement counter-based accumulator	32
Figure 6 – IDD structure.....	33
Figure 7 – Correlator hardware in a channel.....	35
Figure 8 – Normalized acquisition result against Doppler frequency estimate mismatch, for different accumulation intervals, (T_c).....	37
Figure 9 – Sharing a combinational logic block through arbitration and the use of a hybrid QDI-bundled-data interface	39
Figure 10 – Bundled-data DLL	40
Figure 11 – Bundled-data PLL.....	42
Figure 12 – An inverter layout by cellTK	46
Figure 13 – Cell integration.....	47
Figure 14 – Layout of the asynchronous GPS baseband processor.....	48
Figure 15 – Arbitration between channels.....	49
Figure 16 – Power grid reinforcement on metal layers 7 and 8	52
Figure 17 – Closed-up view of power grid reinforcement	53
Figure 18 – Die photo.....	55
Figure 19 – Position accuracy using 6 satellites.....	56
Figure 20 – Tracking sensitivity test	57
Figure 21 – In-phase and quadrature accumulations phasor comparison between the ASIC and the software receiver.....	58
Figure 22 – PLL Doppler frequency tracking performance compared to software receiver	59
Figure 23 – Host-ASIC ecosystem with COTS frontend for testing.....	62
Figure 24 – Test setup for post-silicon verification.....	63
Figure 25 – Accumulator module power from measured test structure for different operation frequencies and supply voltages.....	65
Figure 26 – NCO module power from measured test structure for different operation frequencies and supply voltages.....	67
Figure 27 – C-element for half-timing-assumption illustration	69
Figure 28 – Deadlock seen in detailed SPICE simulation.....	70
Figure 29 – Long wire driven by a weak inverter. Each segment of the scale shown is 50 microns	71
Figure 30 – Post-layout SPICE simulation of the revision without deadlock.....	72
Figure 31 – Effect of the number of fractional bits on leakage and dynamic power of the tracking loops shared by 5 channels, for 40nm, 65nm and 90nm technology nodes	74
Figure 32 – Effect of the number of channels on the power of the tracking loops with 8 fractional bits, for 40nm, 65nm and 90nm technology nodes	76

Figure 33 – Cross section view of a normally-open NEM relay	79
Figure 34 – (a) “NMOS” with normally-open relay (b) “NMOS” with normally-closed relay (c) “PMOS” with normally-open relay (d) “PMOS” with normally-closed relay	81
Figure 35 – Normally-open NEM relay implementation of (a) Inverter, (b) Buffer, (c) NAND, and (d) AND	82
Figure 36 – Effective NEM relay implementations of (a) AND, (b) OR, (c) XOR, and (d) XNOR	83
Figure 37 – N-input C-element implemented with NEM relays	86
Figure 38 – Energy per operation versus delay comparison for C-element with different number input bits, implemented with NEM relays and CMOS to drive a 25fF load	88
Figure 39 – WCHB with 1 bit input (L) and output (R) using NEM relays, staticizers omitted	89
Figure 40 – PCHB 2-bit input and 1-bit output using NEM relays, staticizers omitted	91
Figure 41 – Simulation of the PCHB dataless channel with NEM relays	92
Figure 42 – Optimized output false rail of a 2-input PCHB AND using NEM relays	93
Figure 43 – Average energy per operation at 10MHz for PCHB AND with different number of inputs, implemented with NEM relays and CMOS with 25fF output loads	94

LIST OF TABLES

Table 1 – Simulated power broken down according to modules during acquisition and tracking	60
Table 2 – Comparison with state-of-the-art.....	61
Table 3 – Extrapolated measured power of accumulators and NCOs for 1V, 5.714MHz	68
Table 4 – Comparison between simulated power and measured power derived from test structures	68

LIST OF ABBREVIATIONS

ADC	Analog-to-Digital Converter
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal-Oxide Semiconductor
C/No	Carrier-to-Noise Ratio
CAD	Computer-Aided Design
CHP	Communicating Hardware Processes
COTS	Commercial Off-The-Shelf
CSP	Communicating Sequential Processes
CST	Code Start Time
DLL	Delay Locked Loop
DOP	Dilution of Precision
DSP	Digital Signal Processor
ESD	Electro-Static Discharge
FLL	Frequency Locked Loop
FPGA	Field-Programmable Gate Array
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HCTA	Half Cycle Timing Assumption
HSE	Handshaking Expansion
I/O	Input/Output
LBS	Location Based Services
MEMS	Micro-Electro-Mechanical Systems
NCO	Numerically Controlled Oscillator
NEMS	Nano-Electro-Mechanical Systems
PCFB	Pre-Charge Full Buffer
PCHB	Pre-Charge Half Buffer
PGA	Pin Grid Array
PLL	Phase Locked Loop
PRN	Pseudo-Random Noise
QDI	Quasi-Delay Insensitive

RAM	Random-Access Memory
RF	Radio Frequency
SOC	System-On-Chip
SPICE	Simulation Program with Integrated Circuit Emphasis
UTC	Coordinated Universal Time
VLSI	Very Large Scale Integration
VTC	Voltage Transfer Characteristic
WCHB	Weak-Conditioned Half Buffer

CHAPTER 1 – INTRODUCTION

1.1 Motivation

Our lives are increasingly being impacted by the use of the Global Positioning System (GPS) technology. We rely on GPS to navigate from one place to another, to locate a person or an object, to provide time synchronization to our telecommunication networks and power grids, and many other everyday applications. With the emergence of the smartphone era, GPS technology has exploded into ubiquity. Location-based services (LBS) are becoming an integral part of mobile computing as they not only assist in personal navigation, but also provide personalized mobile advertisement, context-, preference- and location-aware social media features and local search of business listings, events and other points of interest. In fact, according to a market research group, the global LBS market is projected to triple in the next seven years to 1.3 Trillion dollars by 2020 [17].

However, the true potential of LBS today is limited by the high power consumption of existing GPS receiver chips. According to user forums and surveys, continuous GPS operation typically lasts an hour or two before the battery is fully drained [32], [33]. Overheating issues during GPS usage on smartphones are also commonly reported [31]. Not surprisingly, when the smartphone is running a navigation application, the CPU, display, cellular radios, audio, GPS and other peripherals consume power, some continuously and some intermittently, and the continuously-on GPS receiver is indeed a major power hog [34]. Due to the power constraints on smartphones, the GPS function is only turned on frugally by the users

or the applications and thereby unable to provide continuous location updates that are crucial for LBS applications. Besides, the high power consumption of a GPS receiver is also related to its use in weak-signal or urban-canyon environment. Skyscrapers in cities obstruct the GPS receiver's view of the satellites in the sky and introduce multipath and attenuation in the received signal. Consequently, the GPS receiver needs to continuously monitor its signal and integrate over longer correlation periods to maintain lock.

If GPS receivers are low power enough, augmented reality applications on smartphones and wearable computers can be used extensively throughout the day to improve our quality of life. Fitness applications can implement more advanced features over a single charge of the device to continuously track the distance and paths traveled during a hike or a jog and together with other motion sensors, incorporate daily physical activities into the user's overall fitness regime. Moreover, if the GPS receiver on a mobile device can be left on continuously without battery power depletion concerns, the mobile device can be very accurately synchronized to the Coordinated Universal Time (UTC) through the GPS satellite constellation [35] to achieve better security and reliability in mobile financial transactions. Besides, having accurate global timing synchronization also accommodates increased data rates in communication between distributed systems such as a network of mobile or wearable devices, the internet and cellular networks [10]. These are just a few examples that demonstrate the potential of LBS if only GPS receiver chips consume much less power than what they do today! In fact, the European GNSS Agency in their October 2013 GNSS Market Report forecasts that by the 2022, there will be close to 7 billion GNSS devices in the world, with LBS contributing 47% of the cumulative core

revenue over a decade from 2012 to 2022.

1.2 Related Work

A typical GPS receiver consists of an RF front end and a digital signal processor (DSP). The RF front end receives the GPS signal from the satellites, mixes it down to an intermediate frequency, and samples it. The DSP acquires a lock to multiple GPS satellite signals present in the front end samples and tracks deviations in the signals over time. While the DSP tracks variations in the signal, it also extracts information from it that can be used to compute the position and time — the “navigation solution”.

Significant research effort has been devoted to reducing the power consumption of the RF front end, with current designs having power consumption of less than 10mW [4]. However, more work needs to be done to lower the power consumption of the GPS baseband processor. A powerful DSP can perform all baseband processing in software [5]. This approach is highly reconfigurable and easy to develop and debug but it is not suitable for low power applications. An alternative is to use a hardware correlation engine to handle the fast correlation operations and a microprocessor to handle the rest of the signal processing tasks [6]. In applications where the user only requires infrequent position updates, a possible solution is proposed in [7], where the receiver is only powered intermittently. This power-cycling approach relies heavily on rapid re-acquisition and does not continuously track satellites.

This work focuses on addressing the power consumption issue in the baseband

processor when continuous position information is required. Instead of using a single DSP or a correlation engine with software support to perform baseband processing, we designed a hybrid architecture that decouples crucial GPS receiver operations from other post-processing that can either be managed locally by a co-processor or be separately managed by the cloud or base station. Our system performs all baseband processing on-the-fly in hardware, leaving only initialization and navigation solution computation to software. This approach not only provides an optimized hybrid hardware and software solution but is also ideal for applications that need to deploy ultra-low-power mobile GPS receivers that transmit just enough information back to a base station to compute its position and time information. Moreover, our clockless data-flow driven baseband processor can be paired with any conventional GPS L1 RF front end. It is programmable to support different front end sampling and intermediate frequencies and mixing schemes.

The next chapter discusses the GPS signal structure and identifies the asynchrony in it. While the asynchrony in the GPS signal makes building a low power receiver with conventional techniques difficult, it can be exploited to efficiently process the GPS signals using asynchronous circuit design. Chapter 3 gives an overview of typical GPS baseband processor functions and the optimizations that were made for our design are highlighted in Chapter 4. Chip implementation is detailed in Chapter 5. The simulation and measured results are discussed in Chapter 6. Finally, Chapter 7 introduces advanced technologies for our design.

CHAPTER 2 – GPS SIGNAL STRUCTURE

2.1 GPS L1 C/A

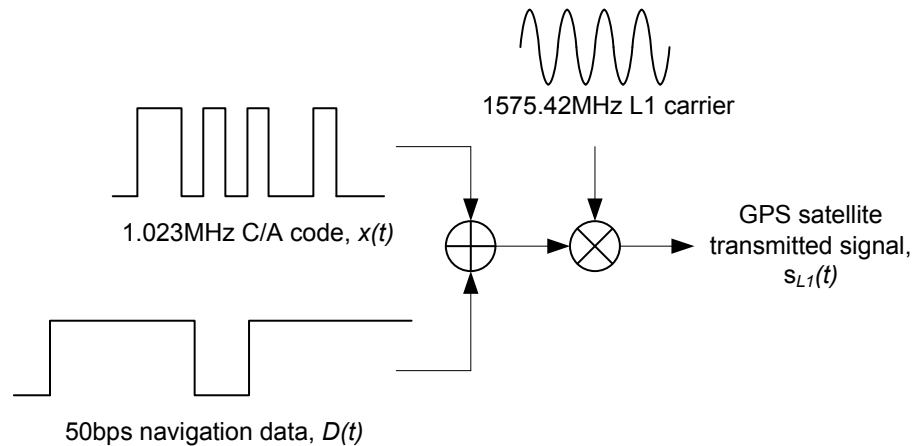


Figure 1 – GPS L1 C/A signal structure

The GPS uses spread-spectrum signaling to modulate a carrier with ranging codes. This technique allows a GPS receiver to use Code Division Multiple Access (CDMA) to uniquely identify the signal from each satellite by the satellite's unique ranging code. Modern GPS satellites transmit signals at the L1, L2 and L5 carrier frequencies with civil and military codes. Since our system is designed to process the L1 civil signal, I will only describe its signal structure here.

The GPS L1 Coarse/Acquisition (C/A) signal consists of the 1575.42MHz L1 carrier signal modulated by the 1.023Mbps C/A ranging code which is in turn modulated by the 50bps navigation message as shown in Figure 1. The C/A ranging code has a period of 1ms and is a special class of pseudorandom noise (PRN) sequence of +1 and -1 chips and each satellite is assigned a unique sequence. The navigation data is a sequence of +1 and -1 that carries satellite orbital information,

satellite time and error correction parameters [8]. The L1 C/A signal transmitted with an average power of P_{L1} from a GPS satellite can be represented mathematically as

$$s_{L1}(t) = \sqrt{2P_{L1}} D(t)x(t) \cos(2\pi f_{L1}t + \theta_{Tx}) \quad (1)$$

D is the navigation data; x is the C/A code and f_{L1} the L1 carrier frequency.

The signal from a particular satellite reaches the receiver's RF front end after some transmission delay. Hence the time of flight of the signals from different satellites introduces different delays clearly seen in the ranging codes. This time-of-flight differences form the fundamentals of radio-navigation on which the GPS system is based. In addition, due to the receiver's front end oscillator error and the relative movement of the satellite and the receiver, a Doppler frequency shift is introduced in the received signal. The receiver needs to accurately estimate and continuously track these uncertainties in the signal in order to compute its navigation solution. To achieve that, the receiver first gets a rough estimate of the transmission delay and Doppler frequency shift for each available satellite through a procedure called acquisition and then refine the estimates through tracking loops. A delay-locked loop (DLL) tracks deviations in the estimate of the transmission delay whereas a frequency-locked loop (FLL) or a phase-locked loop (PLL) tracks deviations in the Doppler frequency estimate.

It is important to realize that the signal in (1), after conditioning by the front end, consists of different clock domains, the fastest of which is the RF front end ADC sampling rate, followed by the $1/1.023\text{e6}$ second PRN code chip interval in x , followed by the 1ms PRN code period in x , and finally the 20ms navigation data rate in D .

2.2 Asynchrony in Baseband Processing

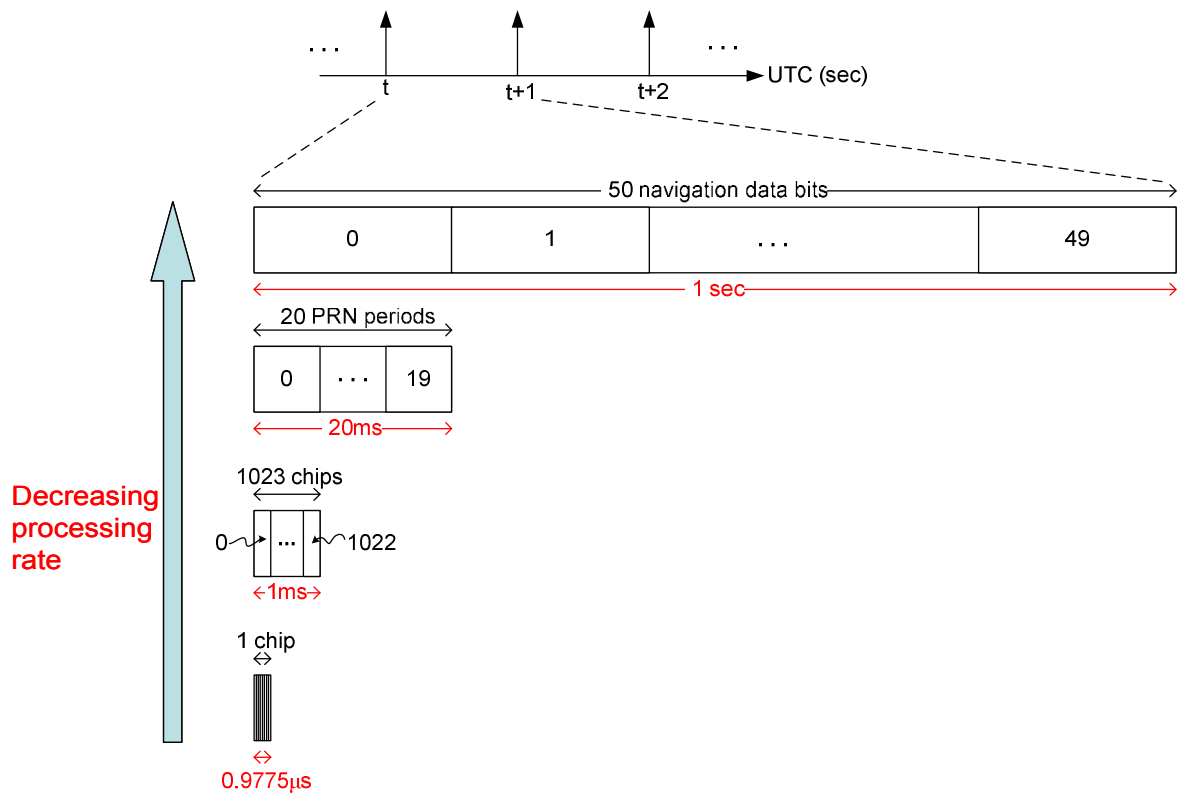


Figure 2 – Baseband processing's attempt to replicate the satellite's atomic clock time through the GPS signal structure

Another way of thinking about the role of the GPS baseband processing is that it needs to track the signal from the satellite and attempts to replicate the satellite's atomic clock time seen in its signal structure. By comparing the satellite's clock time and the receiver's local time, it can find the time of flight of the signal from which position can be computed. The entire GPS satellite constellation can be thought of as being synchronized to a virtual 1 second time stamp in Coordinated Universal Time (UTC). Through the GPS signal structure, each second can be further scaled down all the way to the nanosecond level as shown in Figure 2. Firstly, one second consists of

50 navigation data bits where each data bit consists of 20 PRN code periods which in turn make up 20ms. Each PRN code period consists of 1023 chips which make up 1ms. Each chip spans $0.9775\mu\text{s}$ and can further be divided down to fractions of a chip to give us nanosecond resolution.

The time segments presented here look neatly divided up but in reality, the baseband processor is inherently asynchronous. While the baseband processor attempts to create the satellite clock replicas, it needs to correct for the deviations from these nominal values. These deviations are caused by Doppler frequency shifts that not only affect the carrier frequency but also the PRN code period. In other words, there exists an intra-channel asynchrony issue where the satellite clock replica that the channel is trying to create has deviations; it is trying to make it in synchrony with the satellite clocks. Besides, the baseband processor does not only track one satellite but four or more. Some satellites are closer to the receiver and some further. Although the satellites nominally broadcast simultaneously, their signals arrive at the receiver at different times. Hence the satellite clock replica created in each channel shows different times. This inter-channel asynchrony issue makes access to shared computation resources among channels not as predictable. Thirdly, the baseband processor also faces computation asynchrony. The incoming samples from the RF front end contain fractions of a chip each. The baseband processor integrates the received signal over 1023 chips to account for 1ms, from which it can further count its way to 1 second. There is an immense amount of information coming in from the front end and the data processing rate is the fastest in this workhorse portion of the baseband processor known as the correlators. The code replicas are generated at the Doppler-shifted chipping rate and upsampled to the frontend sampling frequency; the

tracking loops are driven by each channel once every millisecond. Moreover, the navigation data is decoded at a rate from 1 kHz to 50Hz and the phase measurements are made and output every second. A typical synchronous system ends up having to generate clock ratios that are difficult to manage. For example, if a Zarlink GP2015 RF frontend with a sampling frequency of 40/7 MHz is used, clock ratios of 40000000:7161000:7000:350:7 are needed to avoid overclocking the subsystems and consuming unnecessary power.

By embracing and exploiting asynchrony, we open up for ourselves plenty of opportunities for low power operation not only with macro-architecture level optimizations where all subsystems run at their optimal frequencies, but also with micro-architecture level optimizations in subsystems from correlators to tracking loops and all the way to the bit level. Therefore, asynchronous circuits are ideal for low-power designs in such a system where “asynchronous” clock domain crossings are not possible due to inexact timing and where the rate of required throughput varies over time [1], [2], [3].

2.3 Asynchronous VLSI

Instead of using a global time reference to sequence computation in synchronous (clocked) circuits, asynchronous circuits use handshake protocols. Data tokens are passed from one pipeline stage to the next through handshakes. This data-flow driven behavior allows asynchronous systems to naturally support varying data rates and throughput to achieve average-case pipeline latency rather than worst-case latency seen in synchronous pipelines. Hence, all subsystems only run as fast as they

need to run without clocking. Besides, as these handshakes are coordinated locally among communicating processes, asynchronous circuits do not face global clock distribution issues. The equivalent of perfect clock gating is also realized in asynchronous systems since dynamic power is only consumed when the circuit is actually processing data. Two common asynchronous handshake protocols are the bundled-data [12] and quasi-delay-insensitive (QDI) techniques [16]. Since our system uses both of these techniques, I will describe them in further detail. Several other asynchronous design styles are summarized in [29].

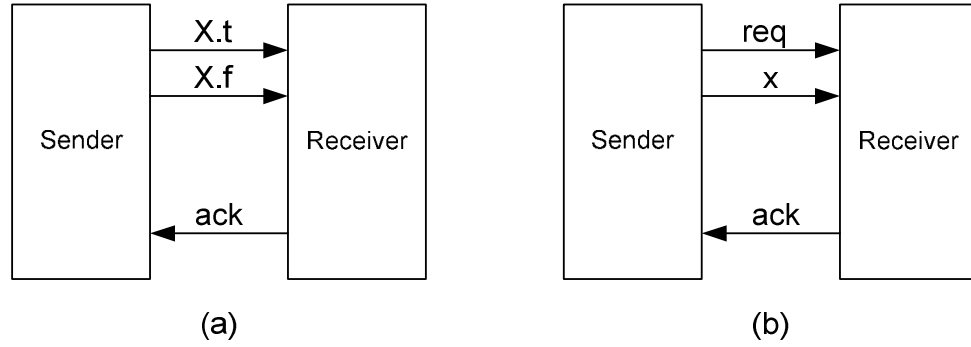


Figure 3 – Two common asynchronous handshake protocols: (a) Dual-rail QDI (b) Bundled-data

2.3.1 Bundled-Data

The bundled-data technique uses the most basic form of asynchronous handshake between two subsystems; as shown in Figure 3(b), a request signal (req) from the sender subsystem is used to indicate that its data (x) is valid and an acknowledge signal (ack) from the receiving end is used to indicate that the data has been received. Since the bundled-data design is single-rail encoded and resembles synchronous circuits, it can be synthesized using commercial synchronous tools.

Importantly, with the bundled-data technique, some timing assumptions must be made about the computation delay in the sender subsystem. To ensure that the request signal is only asserted after the data is valid, the request signal is driven by a delay-line that is longer than the computation delay in the sender subsystem.

2.3.2 QDI

Unlike bundled-data, QDI circuits do not need to make such timing assumptions other than the isochronic fork assumption [16]. QDI circuits have the data validity information encoded in the data itself; one popular encoding is the dual-rail encoding shown in Figure 3(a) where each data bit (X) is represented by two bits: $X.t$ and $X.f$. When the data is valid, one of $X.t$ and $X.f$ is asserted and the value of the data depends on whether $X.t$ or $X.f$ is the one asserted. Such encodings within the data allows QDI circuits to have excellent self-timed properties and the least timing assumptions compared to other asynchronous circuit styles, making them very robust against delay variations from variability in fabrication process and circuit operating conditions. However, such encodings in QDI circuits have area and power overheads, especially for wide datapaths. As we will see in later chapters, because of these reasons, we opted for bundled-data instead of QDI in the math functions of the GPS tracking loops.

The CHP language is a variant of CSP and is widely used when describing QDI asynchronous circuits [1]. The QDI system is first described in CHP and then, through the process of Martin synthesis [1], translated to handshaking expansions (HSE) and subsequently transistor-level circuit implementation. Some of the more

commonly used QDI circuit templates are the Weak-Conditioned Half Buffer (WCHB), Pre-Charge Half Buffer (PCHB) and Pre-Charge Full Buffer (PCFB) [30].

CHAPTER 3 – SYSTEM OVERVIEW

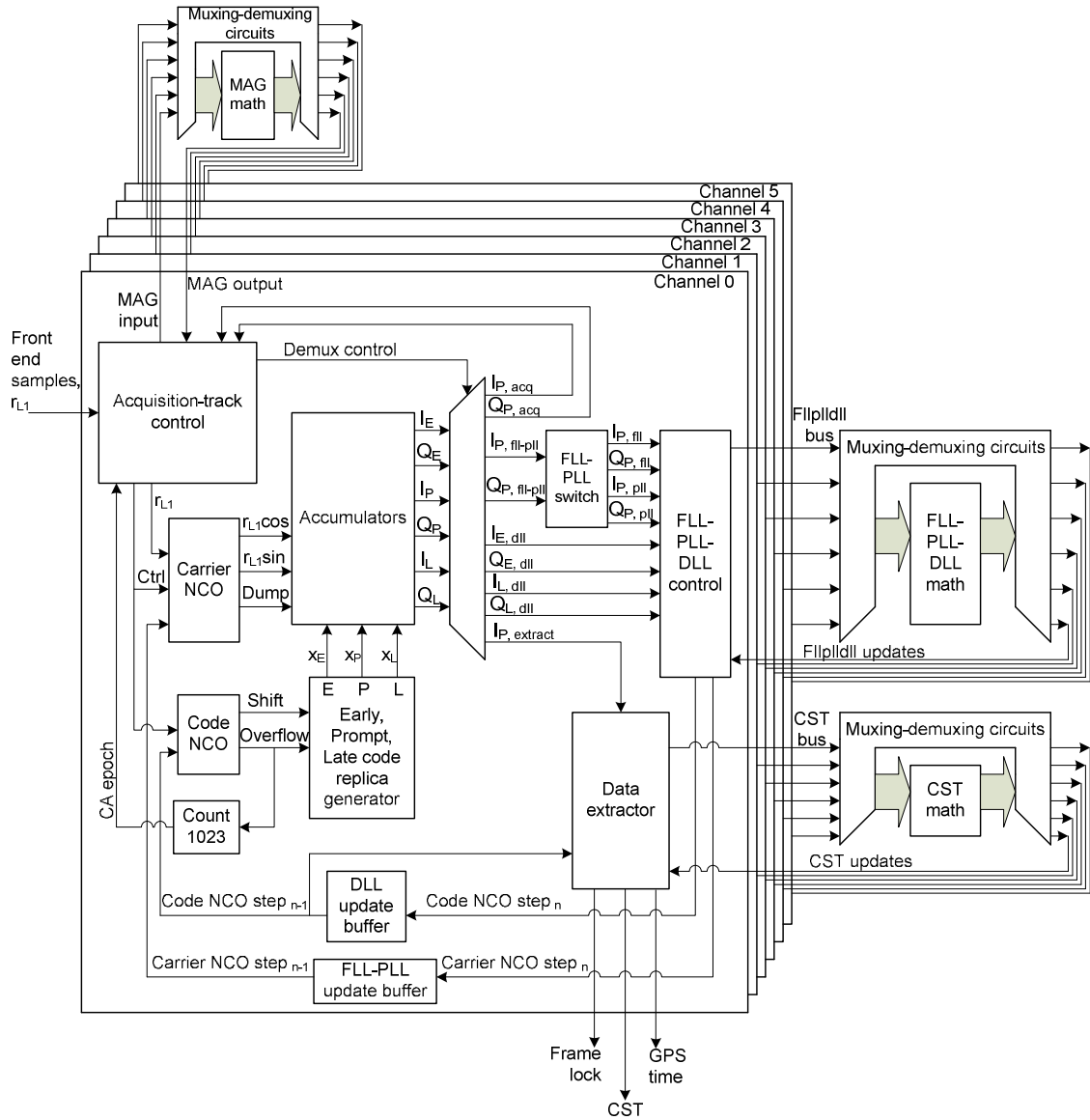


Figure 4 – System diagram

Our dataflow-driven system expects 1-bit samples from an RF front end which it uses to acquire, track, extract crucial data and output measurements at 1Hz to a co-processor or base station for navigation solution computation. In order to solve the

four unknowns corresponding to the receiver time and location coordinates, a GPS receiver needs to track at least four satellites. To allow for redundancy and the flexibility to compute over-determined least-squares solutions, our system tracks six satellites with six channels where each channel is responsible for processing the signal from one particular satellite. The system is optimized to have all channels share a single tracking loop. A more comprehensive description of a typical GPS baseband processor can be found in [9], [10], [11].

3.1 Acquisition

Before a receiver can begin to track a satellite, it needs to know which satellite to track, and an estimate of the Doppler frequency and code offset of the signal for that particular satellite. Therefore, during signal acquisition, a typical receiver searches the expected Doppler frequency space and code offset space of candidate satellites. First, for each candidate satellite, the input signal from the RF front end undergoes a carrier wipeoff with a candidate Doppler frequency \hat{f}_D and a code-wipeoff using a locally generated code replica, X_P , with a candidate code offset, $\hat{\tau}$. This candidate hypothesis consisting of \hat{f}_D , X_P and $\hat{\tau}$ is tested by correlating the anticipated signal with the received signal over a 1ms code period (N front end samples) to yield an in-phase accumulation, I and the quadrature accumulation, Q . Mathematically, I and Q can be written as:

$$I = \sum_{k=1}^N r_{L1}(t_k) X_P(t_k - \hat{\tau}) \cos \left\{ 2\pi (f_{IF} + \hat{f}_D) t_k \right\} \quad (2)$$

$$Q = \sum_{k=1}^N r_{L1}(t_k) X_P(t_k - \hat{\tau}) \sin \left\{ 2\pi (f_{IF} + \hat{f}_D) t_k \right\} \quad (3)$$

The k^{th} sample from the RF front end is $r_{L1}(t_k)$, and the intermediate frequency to which the satellite signal has been mixed down by the RF front end is f_{IF} . If the correlation power $I^2 + Q^2$ exceeds a threshold, then the candidate hypothesis has been validated and acquisition for that particular satellite has been successfully achieved. The system implements an assisted acquisition approach where it is provided with the satellite PRN number to acquire as well as their respective correlation power threshold and slowly-varying Doppler frequency estimate \hat{f}_D . This feature will be explored further in the Optimization chapter. In each channel in Figure 4, two 32-bit numerically controlled oscillators (NCO) are driven by the data-flow from the front end. The Code NCO controls the chipping rate of the code replica generator. Every time the Code NCO overflows, a new code chip is generated and 1023 overflow occurrences of the Code NCO mark the end of a 1ms code period, also known as an accumulation interval. Whenever a data bit is available from the front end, the Carrier NCO mixes it with its sinusoids to generate 3-bit outputs for the in-phase and quadrature components. They are then mixed with the code replica to form the correlator summands in (2) and (3).

3.2 Tracking

After acquisition, the receiver channel has enough knowledge of the code

offset and Doppler frequency to roughly align its code and carrier replicas to the received signal. The receiver channel now enters tracking mode to continuously adjust the alignment of its signal replicas to the received signal. Besides the Prompt correlators that produce the prompt in-phase accumulation in (2) and quadrature accumulation in (3), each channel also uses Early correlators with the replica PRN code advanced by $1/2$ chip to produce the early in-phase and quadrature accumulations, and likewise, Late correlators with the replica PRN code delayed by $1/2$ chip to produce the late in-phase and quadrature accumulations. These early and late accumulations are needed to implement a DLL which adjusts the chipping rate of the code NCO to correct misalignments in phase between the replica PRN code and the received PRN code.

Other than tracking the code phase, the system uses an FLL and a PLL to track the Doppler frequency. The FLL is more robust to noise, has better dynamic performance and has a wider pull-in range than the PLL but its measurements are noisier. Hence, the FLL is only used to obtain a more accurate estimate of the Doppler frequency momentarily after acquisition before handing the task over to the PLL. This transition happens after a programmable delay (400ms used in testing) to ensure that the FLL has already converged. The PLL locks the phase of the in-phase portion of the carrier replica to the received signal. It is important to realize that the DLL, FLL and PLL tracking loops are only called into action at the end of an accumulation period which occurs once every 1ms in each channel. The subsequent Optimization chapter will give more details on the tracking loops algorithms.

3.3 Data Extraction

As seen in (1), the GPS satellite's signal contains a 50bps navigation data stream. This data stream is modulated with the satellite's PRN code which has a period of 1ms. As a result, 20 periods of the satellite's PRN code contain information for the same data bit. Since a PLL is used for tracking, the accumulation power is concentrated in the in-phase accumulation. Hence, at the end of a 1ms accumulation period, one raw data bit is extracted from just the sign of the prompt in-phase accumulation. To remove the redundant data bits, the raw data bits are down-sampled by 20 to 1. From the down-sampled data bits, the system attempts to achieve frame lock. A preamble bit sequence of "10001011" marks the start of a data message frame. Since the PLL discriminator has a 180-degree phase ambiguity, the extracted data bits may be inverted. Thus, a search for both the normal and inverted preamble bit sequence is required. To ensure that the detected bit sequence is indeed the preamble bits instead of a string of navigation data message bits that happens to resemble the preamble bit sequence, the system extracts the GPS satellite time information from the data message to check if it correctly increments from one subframe to the next.

The system uses a code accumulator for each channel, hereafter called Code Start Times (CST), as time stamps in units of sample counts to mark the start of the C/A PRN code period. Every time the DLL updates (approximately 1ms), the CST is incremented by the number of samples in that approximate 1ms time lapse. At an interval of 1s, determined by counting 1000 raw data bits from the start of a subframe, each channel reports its frame lock status, GPS satellite time and its corresponding 64-

bit CST time stamp. With just these three pieces of information, an external co-processor or a base station can compute the navigation solution. Firstly, relative pseudoranges are computed from the CST values of each channel using the first channel as the reference [9]. From the relative pseudoranges and corresponding GPS times, the navigation solution can be computed [9], [10].

CHAPTER 4 – OPTIMIZATIONS

4.1 Natural Frequency Operation of All Sub-systems

The structure of the GPS signal allows for signal processing at different rates. By designing the system wholly with asynchronous circuit techniques, we can enable each sub-system – fast and slow – to run at its natural frequency. Tokens are passed down through handshakes from fast sub-systems to slow sub-systems to enable on-the-fly signal processing without the use of memory. The correlators comprising accumulate and dump modules, code and carrier NCOs and their controls operate at the RF front end sampling frequency of several MHz. This is the fastest rate the system is expected to handle followed by the Doppler-shifted code chipping rate and the next signal processing step involving the tracking loops is three orders of magnitude slower because each channel's correlators only dump their accumulations at the end of a 1ms accumulation period. Furthermore, only 1 in every 20 raw data bits extracted from the sign of the 1 kHz stream of prompt in-phase accumulations is used for navigation message extraction. From the 50bps data bit stream, the system derives frame lock and satellite time information and finally outputs CST measurements at 1Hz. Other than the complicated fixed-point arithmetic circuits implemented with bundled data technique, all modules in the system are implemented using QDI techniques to ensure timing robustness.

4.2 Adder and Increment-Decrement Accumulators

Accumulators operate at the RF front end sampling frequency and are one of

the more power-hungry components in the whole system. Moreover, there are six accumulators per channel to produce early, prompt and late pairs of in-phase and quadrature accumulations. The 3-bit accumulator inputs which are represented by the summands in (2) or (3) and their early and late variants, are accumulated over one accumulation period to produce 16-bit sum outputs. Sixteen bits are enough to represent the accumulation sum in 2's complement. When the system is tracking well, the iterative cross-correlation of the replica and received signals as shown in (2) and (3) will either add up towards $-\infty$ or $+\infty$ depending on the sign of the encoded navigation data bit. Besides that, as the 3-bit summand is iteratively added to the 16-bit sum, the bits of higher significance do not switch frequently. Based on these observations, a constant-time increment-decrement-based accumulator design is introduced to reduce transistor switching activity and hence the dynamic power consumption of this critical module. Instead of using a standard 16-bit accumulator which consists of an adder with its output fed back to one of its inputs via a register, I implement a standard 3-bit accumulator coupled with an increment-decrement-dump unit (IDD). The reason for this 3-bit and 13-bit split is that the inputs are 3 bits and they can be positive or negative to add or subtract from the intermediate accumulator value. Hence, only a minimal number of low order bits (three in this case) are required to cycle in and out of the adder in the standard 3-bit accumulator. The rest of the higher order bits are handled by the IDD.

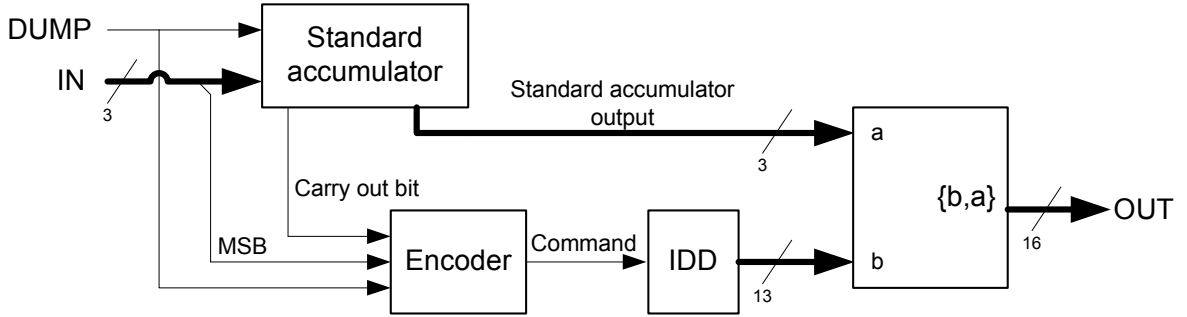


Figure 5 – Increment-decrement counter-based accumulator

In Figure 5, an encoder issues the command to increment or decrement based on the polarity of the input summand and the carry out from the adder in the standard accumulator. If the input to the accumulator is positive, the encoder only issues an increment command if there is a carry out from the standard accumulator; otherwise, it issues no command and hence triggers no transistor switching activity at all in the entire IDD. Note that the negative of a 2's complement number a ($a > 0$) represented with n bits can be written as

$$-a = 2^n - a \quad (4)$$

For example, if a is 1, then 7 and -1 represented as a 3-bit 2's complement number yield the same result. A “borrow” is obtained from the virtual $(n+1)^{\text{th}}$ bit. Hence, if the input to the accumulator is negative, the encoder only issues a decrement command if there is no carry out from the standard accumulator which would otherwise cancel out the “borrow” loaned from the higher order bits in the IDD.

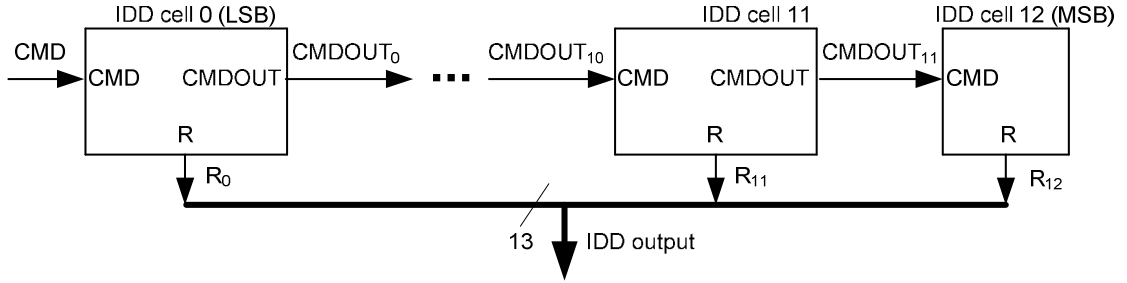


Figure 6 – IDD structure

I design the constant-time 13-bit IDD with 13 cells, each of which implements a need-based command propagation structure. If the IDD receives a command to increment, the LSB cell will increment its own value and will only propagate the increment command to the next higher-order cell if the LSB cell has a carry out; otherwise the higher-order cells experience no switching activity at all. Other than the MSB cell which does not need to propagate any commands further, all higher-order cells have the same need-based command propagation behavior as the LSB cell. Likewise, each IDD cell only propagates the decrement command if it needs to “borrow”. The DUMP command which the accumulator receives once only every 1ms, is propagated throughout the IDD from the LSB to the MSB cell. The full accumulation result is just a concatenation of the result from the standard 3-bit accumulator and the result from the 13-bit IDD.

From SPICE simulation, the power consumed by a naïve 16-bit accumulator is about 40μW whereas that for the system’s optimized IDD accumulator is about 10μW, a power savings of 4X.

4.3 Memoryless Code Phase Acquisition

Before a GPS receiver can start tracking, it needs to know what satellites to track, their code offsets, and their Doppler frequencies. Due to the computation burden involved in a full acquisition procedure, it is typically done either with fast Fourier transforms or with a bank of correlators. However, the receiver does not need to perform acquisition frequently other than during initialization or when it loses lock on the satellites. Moreover, for most mobile personal navigation applications that this dissertation covers, the receiver does not have to deal with high-dynamics and the Doppler frequencies do not vary rapidly. The maximum rate of change in Doppler frequency for a stationary receiver on Earth is about 0.8Hz/s and the movement of the receiver adds about 1.5Hz/s for every km/h [15]. Since the Doppler frequencies estimates are relatively stable, they can be handed off together with the respective satellite PRN number from software in the host system. The code phase however changes more rapidly and requires stricter synchronization between the host, the frontend and the mobile baseband processor ASIC for proper initialization.

Based on the reasons above, we introduced an asymmetric acquisition scheme that allows each channel to be initialized from the host with the satellite PRN number and Doppler frequency estimate so that each channel can then acquire their own code phase offset and subsequently transition to tracking. With such an acquisition scheme, the receiver does not need a dedicated bank of correlators for full acquisition but can re-use the same correlators used for tracking, resulting in minimal hardware footprint on a baseband processor ASIC optimized for low-power tracking. Figure 7 shows the same correlator hardware used for acquisition as well as tracking where a

demultiplexer routes the accumulation results to the threshold detector during acquisition or to the tracking loops in tracking mode. The threshold detector computes the acquisition result $I^2 + Q^2$ where I is the in-phase accumulation and Q is the quadrature accumulation. Initialized with a Doppler frequency estimate and the satellite to track, each channel looks for the correct code offset by iteratively performing 1ms-long accumulations and dropping a frontend sample and incrementing an offset counter at the end of each accumulation period, until the acquisition result exceeds a given threshold. The number of samples skipped is the code offset.

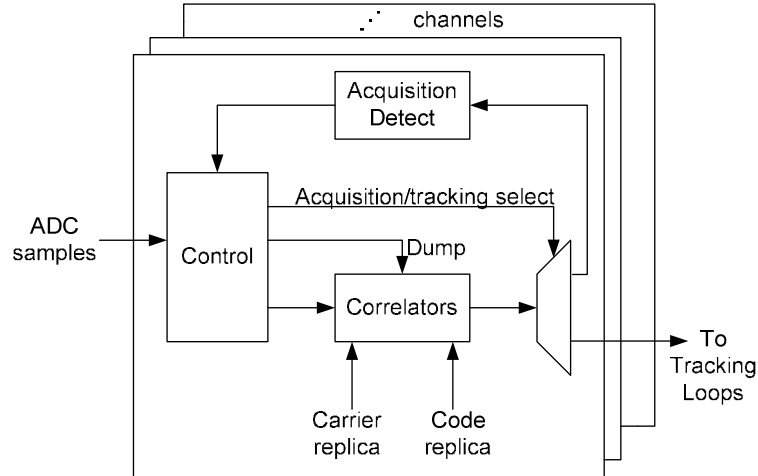


Figure 7 – Correlator hardware in a channel

One obvious drawback for a serial search like this is that the worst-case acquisition time is long. For an X MHz sampling frequency, the worst-case time needed to acquire the code offset is approximately X seconds. Besides that, the 1ms accumulation interval in this basic implementation does not provide enough correlation gain to perform well in weak-signal environment or to be robust against a large Doppler frequency estimate mismatch. Standard textbook techniques can be

used for better acquisition and they are discussed below.

To improve the correlation gain, the accumulation interval, T_c , can be increased. Referring to Figure 7, this can easily be implemented by delaying just the Dump signal. Increasing the accumulation interval becomes especially helpful in weak-signal conditions but there is a risk of having navigation data bit transitions in the interval which will degrade the accumulation result. Another disadvantage is that doubling the accumulation interval doubles the time needed to acquire, resulting in longer acquisition times that might not be suitable for some applications. The issue can be solved by increasing the number of frontend samples to skip over at the end of each accumulation interval. Instead of just 1 sample in the basic implementation, 2 samples can be skipped over to half the acquisition time, 3 samples to reduce the time to a third and so on. For an X MHz sampling frequency, 1 frontend sample is just a fraction corresponding to $1.023/X$ chips and the typical code search space resolution is half a chip.

The asymmetric acquisition scheme relies on a Doppler frequency estimate provided for each channel during system initialization. That Doppler frequency estimates do not need to be very accurate for this acquisition scheme to work allows for a less restrictive system initialization procedure. The Doppler estimates can be computed separately in a local or remote host and be delivered through a wired or wireless communication link. The velocity of the receiver, if not taken into account by the remote host, introduces Doppler rate of change to the extent of about 1.5Hz/s for every 1km/h. The local oscillator in the receiver also introduces Doppler frequency errors but can be accounted for if the local oscillator in the receiver is

synchronized to the remote host [15]. The mismatch in the Doppler frequency estimate ($\Delta f_{Doppler}$) from the true value causes the acquisition result to fall off as a sinc function. The effect of a mismatch in Doppler frequency estimates on the acquisition result is tested by running the asymmetric acquisition algorithm on satellite signal data generated from the Spirent GNSS signal simulator for normal signal strength of -130dBm. The result of this test is shown in Figure 8. Even with an accumulation interval of 1ms, successful acquisition can be obtained with a Doppler frequency mismatch of about 200Hz. By increasing the accumulation interval, we increase the correlation gain for small frequency mismatch but the acquisition result falls off more quickly for larger frequency mismatch. Therefore, before initializing the mobile receiver, the system host needs to make the necessary trade-offs depending on satellite signal strengths, acquisition time requirements, losses from the frontend and the host's ability to estimate the Doppler frequencies. Such improvements make the asymmetric acquisition scheme more robust in moderately weak signal conditions.

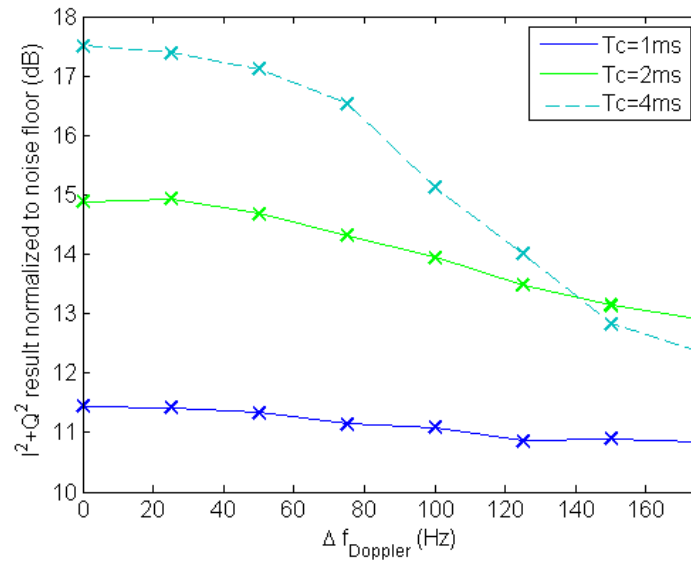


Figure 8 – Normalized acquisition result against Doppler frequency estimate mismatch, for different accumulation intervals, (T_c)

4.4 Bundled-Data Math

Each channel needs to be able to carry out the math functions involved in the FLL, PLL and DLL tracking loops and in CST computation. The arithmetic circuits involved have wide datapaths and are large. For example, the size of the DLL circuit is about 20% of the size a channel. Although these arithmetic functions are highly complicated, each channel only performs such computations to update its system state estimates once every 1ms. Based on these observations, these math functions are implemented in combinational logic blocks using fixed-point arithmetic. As such, the circuit complexity can be reduced and design time saved with commercial logic synthesis tools. Hence, it is obvious that bundled-data is preferred over QDI for these math functions. All channels share these blocks using muxing and de-muxing circuits with arbiters. Data is communicated back and forth between the asynchronous QDI modules and the combinational logic blocks using a hybrid QDI-bundled data interface.

Figure 9 illustrates how six receiver channels share a single combinational logic math block. Through arbitration, the muxing circuit sends data from the asynchronous QDI domain through its dualrail channel M into the combinational logic math block. The true rails of channel M drive the single-rail inputs of the combinational logic block. The arbitration control signal derived from the validity of M drives a matched delay line larger than the computation delay of the combinational logic block. Using asymmetric C-elements, the normal and inverted forms of the outputs from the combinational logic block are gated with the output of the delay line and the acknowledge of the output dualrail channel P to generate the requisite true and

false rails. Once the signal is back in the asynchronous QDI domain, the data is distributed to the corresponding receiver channel through de-muxing circuits.

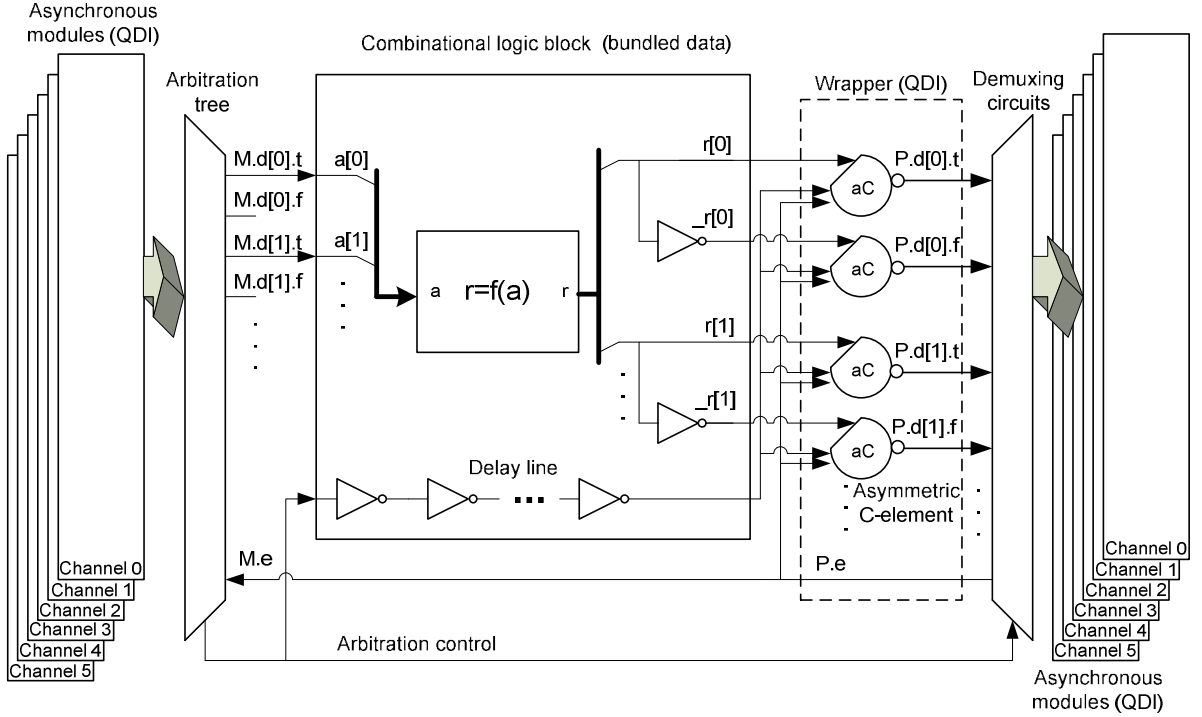


Figure 9 – Sharing a combinational logic block through arbitration and the use of a hybrid QDI-bundled-data interface

4.5 Optimized Tracking Loops

At the end of a 1ms accumulation period, a channel dumps its accumulations and uses the shared FLL, PLL and DLL tracking loops to update its code and carrier NCOs step sizes. Due to the complexity of the math involved, the tracking loops are naturally slow. The slowness of the tracking loop indirectly affects the entire asynchronous system because the NCOs will not obtain their step size updates fast enough to process the next data sample received from the RF front end. To meet the system throughput requirements, the channel defers the application of the tracking

loops outputs to the next accumulation period. In other words, at the end of the n^{th} accumulation period, the NCOs update their step sizes immediately with the tracking loops outputs computed from the $(n-1)^{\text{th}}$ accumulation period. By doing so, we not only can afford to implement slower and more power efficient tracking loops, but also can afford to allow all channels to share the same tracking loops. The tracking loops parameters are analyzed and tuned accordingly.

Besides that, since floating point math consumes more power and increase circuit complexity, I implemented all math circuits in the bundled-data blocks with fixed point arithmetic. The bit widths of the computation sub-blocks are designed in such a way that the transformation does not degrade the final position accuracy significantly.

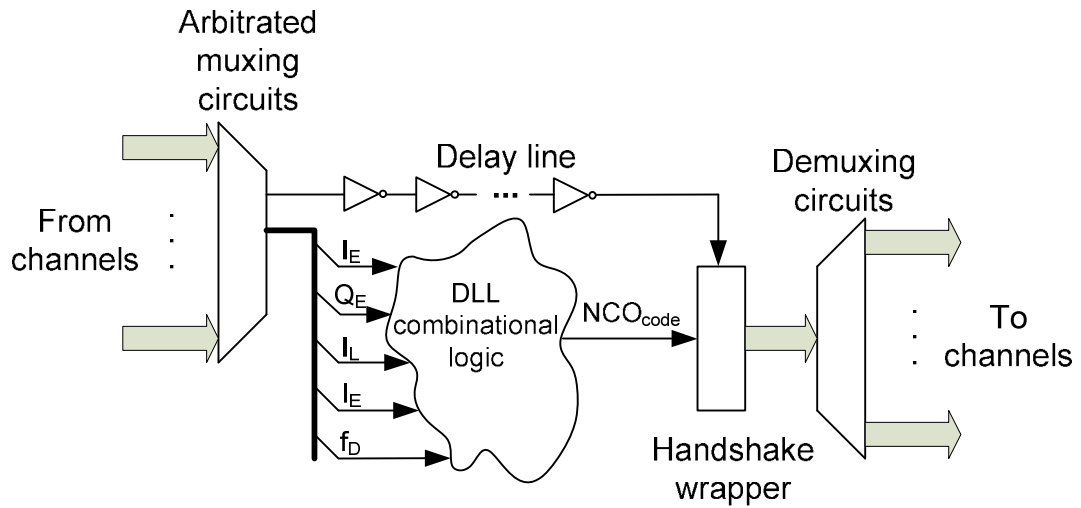


Figure 10 – Bundled-data DLL

The DLL implements a non-coherent normalized Early-Minus-Late (EML) discriminator that computes the delay error estimate from the early and late accumulation results. The discriminator which gives a delay error estimate, in units of

chips, at the end of the n^{th} accumulation period, in the form

$$\tau_{err,n+1} = \frac{1}{2} \left(\frac{\sqrt{I_{E,n}^2 + Q_{E,n}^2} - \sqrt{I_{L,n}^2 + Q_{L,n}^2}}{\sqrt{I_{E,n}^2 + Q_{E,n}^2} + \sqrt{I_{L,n}^2 + Q_{L,n}^2}} \right) \quad (5)$$

where $I_{E,n}$ and $Q_{E,n}$ are the early in-phase and quadrature accumulations whereas $I_{L,n}$ and $Q_{L,n}$ are the late in-phase and quadrature accumulations. The discriminator drives a carrier-aided, first-order loop filter [11] to produce an updated chipping rate $\hat{f}_{chip,n+2}$ for the code replica as shown in (6).

$$\hat{f}_{chip,n+2} = 1.023 \times 10^6 \left(1 + \frac{\hat{f}_{D,n+2}}{f_{L1}} \right) + H \tau_{err,n+1} \quad (6)$$

The Doppler frequency estimated by the FLL or PLL is $\hat{f}_{D,n+2}$ and f_{L1} is the L1 carrier frequency. The gain H of the filter used was about 3. The chipping rate in turn is used to compute an updated step size to the code replica's 32-bit NCO. As shown in (5), the discriminator involves computationally expensive square and square-root operations. To reduce the complexity of this vector magnitude operation, a modified version of the Robertson approximation [13] is applied, where

$$A = \sqrt{I^2 + Q^2} \approx \max \left(\left| I \right| + \frac{1}{4} \left| Q \right|, \left| Q \right| + \frac{1}{4} \left| I \right| \right) \quad (7)$$

Though the largest approximation error is about 10% at a phasor angle of 45 degrees, the actual practical approximation error is less than 3% because the system's PLL drives the phasor angle to zero. While the vector magnitude approximation error directly factors into the DLL discriminator, its effect on chipping rate estimate is insignificant due to mitigation by the loop filter.

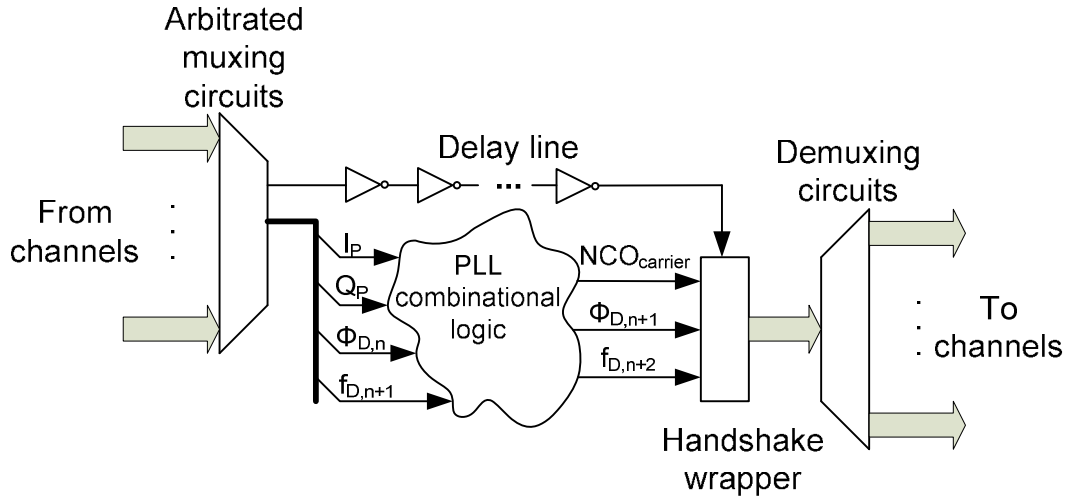


Figure 11 – Bundled-data PLL

The PLL on the other hand uses a two-quadrant arctangent Costas Loop discriminator, ϕ_D . It maximizes the in-phase correlation power while minimizing the quadrature correlation power, which is equivalent to minimizing the phasor angle, $\phi_{D,n+1}$ of the quadrature accumulation vector from the in-phase accumulation vector at the end of the n^{th} accumulation period. A second order loop filter is used to produce an updated Doppler frequency estimate which is in turn used to adjust the step size of the carrier replica's NCO. Figure 11 shows that the PLL is implemented in a similar way to the DLL. However, unlike the DLL, the PLL requires information from the previous iteration; for each channel the phase and Doppler frequency variables from the previous iteration are needed to compute updates for the current iteration. Since all channels share the same PLL combinational logic block, the phase and Doppler frequency variables are bundled into the input and output buses to correspond to their respective channels through the arbitrated muxing and demuxing circuits.

$$\phi_{D,n+1} = \tan^{-1} \left(\frac{Q_{P,n}}{I_{P,n}} \right) \quad (8)$$

The arctangent function in (8) is simplified with a Taylor series small angle approximation where $\tan^{-1}(\theta) \approx \theta$. This approximation is valid for our application because the discriminator values are small and that tiny carrier phase errors alone only correspond to centimeter level errors in the navigation solution.

It is important to note that when there is no computation to be done, the asynchronous tracking loops stay idle and consume no active power. Furthermore, these hardware tracking loops have programmable tuning parameters that can be re-configured from RAM at power-up and be tailored to any frontend frequency plan.

CHAPTER 5 – CHIP IMPLEMENTATION

First, the GPS baseband processor system is designed in the CHP language. The CHP simulation is verified against a GPS software receiver written in MATLAB. This high-level CHP description is then translated into a gate-level description of the system through the process of Martin synthesis [1]. At the time of layout, there are about 300K transistors per receiver channel and about 150K of which are in the fast-rate modules; on the other hand, the shared bundled-data math blocks altogether consists of about a million transistors. The correctness of the gate-level implementation is verified with a co-simulation involving Synopsys VCS and an in-house asynchronous circuit simulator. The detailed simulation results – sequences of values transmitted on all communication channels in the system – from the gate-level co-simulation are found to match those of the CHP simulation. With the netlist in place, we then proceeded with the layout of the entire system.

5.1 Standard and Non-Standard Cell Flow

Commercial tools are not optimized for layout of asynchronous circuits. Typically, asynchronous circuit designers either have to do layout manually or constrain their designs to only a small subset of asynchronous logic families that can be synthesized from their standard cell libraries. Since the asynchronous GPS baseband processor has over 3 million transistors, manual hand layout is not a viable option.

Though the bundled-data math blocks have a combined transistor count of

about a million and have complicated arithmetic circuits including multipliers and dividers, they were implemented more easily than QDI circuits. The bundled-data blocks consist of the same combinational logic gates that conventional synchronous circuit designers are familiar with. However, because the foundry cannot provide the standard cell libraries for logic synthesis and layout, I had to modify conventional commercial tool flows. A standard cell library containing basic combinational logic gates with different drive strengths is created. The logic gates are functionally verified and SPICE-tested in Cadence Virtuoso which are subsequently performance-characterized using Cadence ELC and compiled into a standard cell library using Synopsys tools. With such a library for logic gates, the bundled-data blocks can then be synthesized into a gate-level netlist with conventional tool flows. Typically, once a gate-level netlist is created, circuit designers can feed it into an automated layout tool such as Cadence Encounter to map pre-laid-out standard cells to the netlist and then complete routing. However, without the pre-laid-out standard cell library from the foundry, the same layout problem as QDI circuits surface. Hence commercial tools flows have to be tweaked to support a non-standard cell flow. An in-house tool, known as cellTK [18], can generate layout for any arbitrary transistor netlists. The layout for a small cluster of transistors is created using a custom placer and is then fed into commercial CAD tools as “standard cells” to be integrated into a larger design. For example, Figure 12 shows a simple layout of an inverter created by cellTK. The cells all have power and ground tracks lining the top and bottom edges so that they can be easily integrated into global power and ground rails. A larger design with many cells integrated is shown in Figure 13. The cells are placed between alternating horizontal tracks of power and ground signals. Alternating vertical power and ground

tracks are also added for better power distribution. With cellTK, doing full-custom layout for the QDI circuits and the bundled-data portion is avoided.

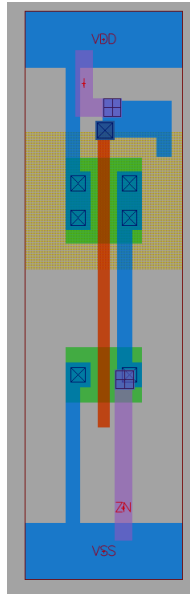


Figure 12 – An inverter layout by cellTK



Figure 13 – Cell integration

The layout of the entire GPS baseband processor is shown in Figure 14. It has a dimension of 4mm x 5mm. Floor-planning and wire-planning are done extensively to ensure that the allocated die area is used efficiently. Each channel is divided into three segments; the first has the accumulators, NCOs and acquisition controls; the second contains controls for the tracking loops and the third segment houses the data decoding sub-system. Each channel consists of about 300K transistors, roughly split equally among the three segments. The layout for each of these segments is generated by cellTK as a whole and then assembled to form a channel. The layout for the channel is then duplicated to create six channels. The bundled-data blocks sit between the forth and the fifth channels and the arbitrated muxing and demuxing circuits are

arranged between the channels as shown in Figure 15 .

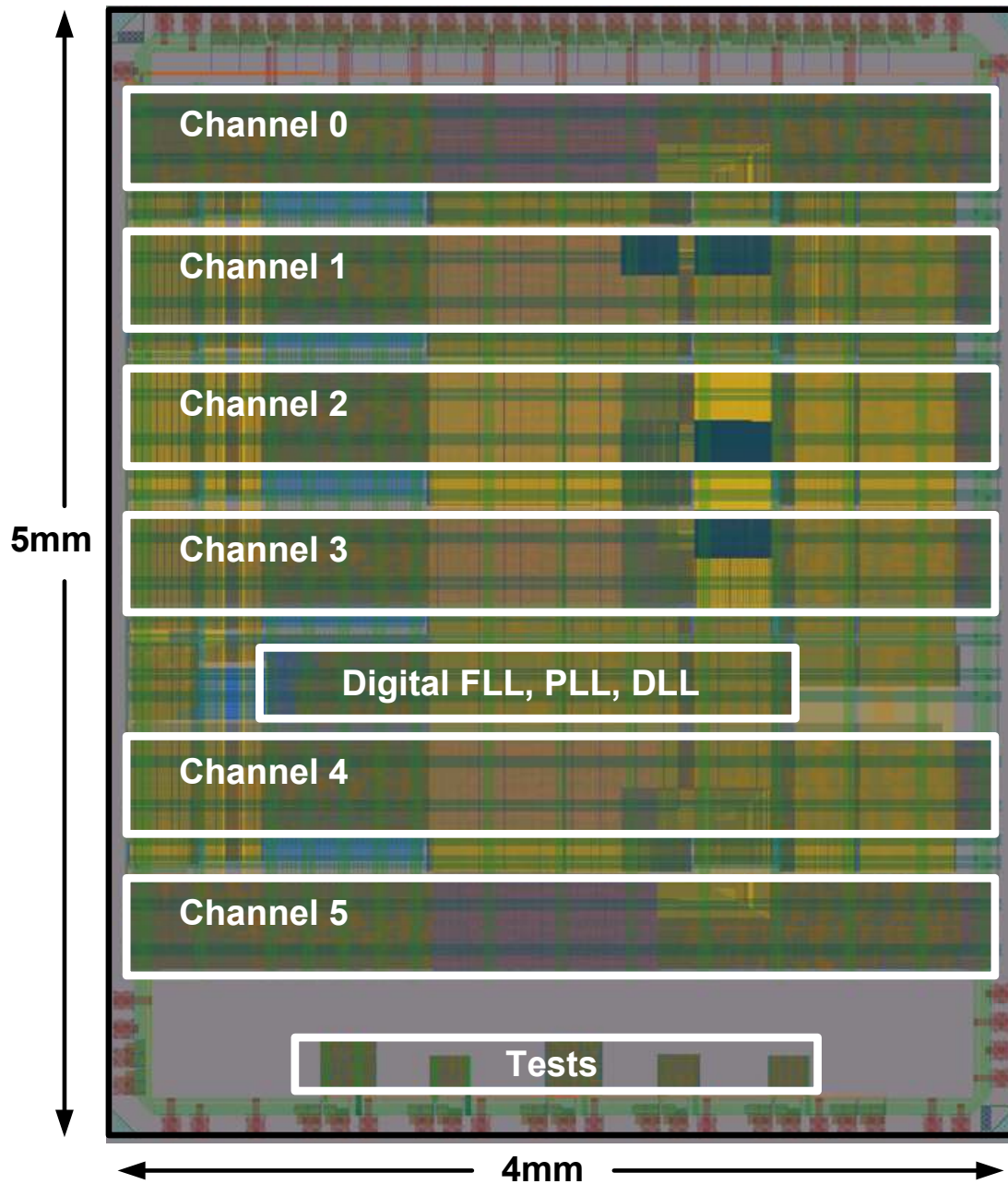


Figure 14 – Layout of the asynchronous GPS baseband processor

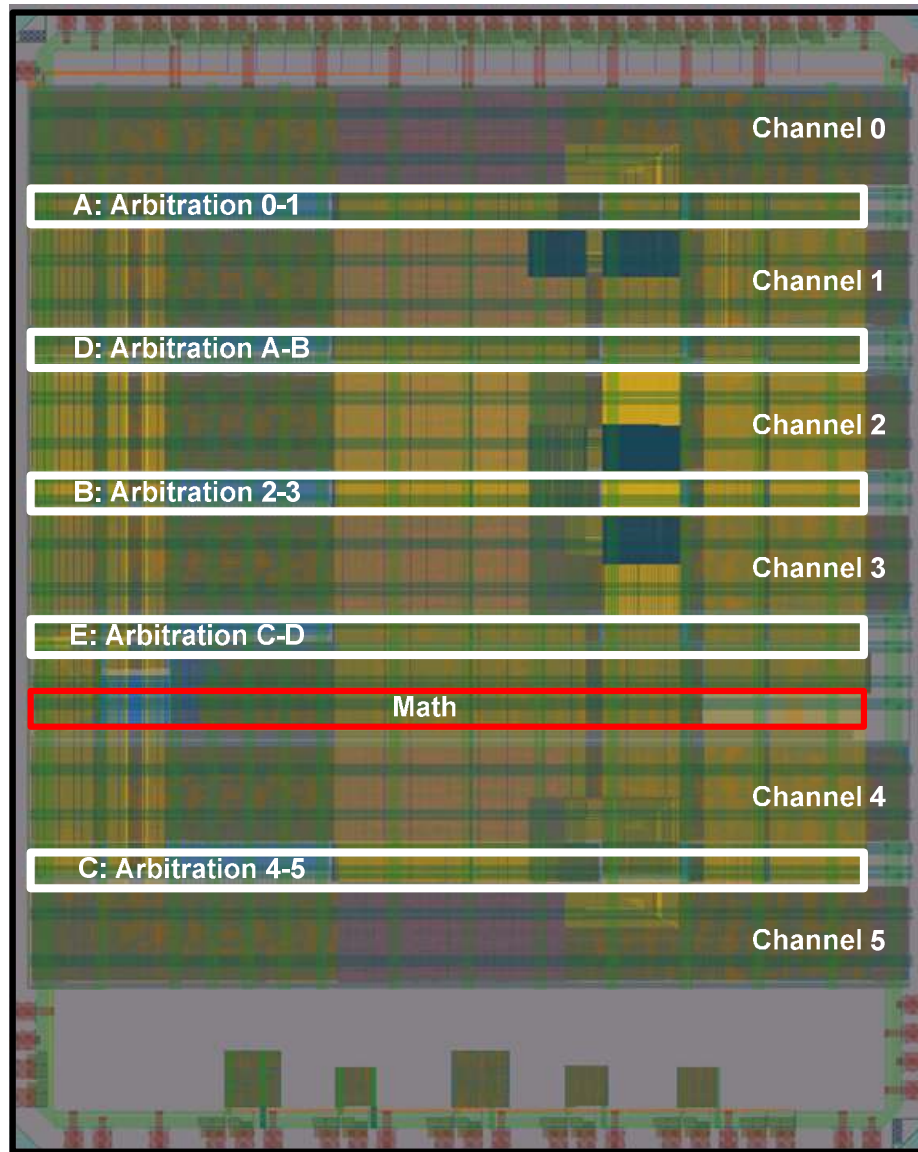


Figure 15 – Arbitration between channels

The arbitration tree arranged in a binary tree structure for 6 channels is not as symmetric as for, say, 8 channels. Three stages of arbitration are needed for 6 channels. Referring to Figure 15, the first stage consists of channels 0 and 1 interfacing with arbitration A, channels 2 and 3 with arbitration B, and channels 4 and 5 with arbitration C. The second stage links A and B to D and finally the third stage

interfaces C and D to the shared math blocks.

5.2 Metal Layer Planning

The system's 90nm process technology allows routing on up to 8 metal layers. Metal layers 1 and 2 are mostly used for local routing within standard cells. Layers 3 and 4 are used for longer-distance routing among standard cells. Metal 5 is used for routing among the arbitration tree circuits. Metal 6 is not only used for routing through congested areas of the layout but also for the global reset signals. Metal layers 7 and 8 are used for global routing of power and ground signals.

To ensure efficient use of Metal 5 wiring area across the arbitration tree, track-sharing is implemented. Short distance routing in the first stage of the tree structure shares the same tracks. Longer distance routing for the last two stages avoid overlaps through staggered placement of input or output pins between the stages.

Besides that, bit-pitch is carefully planned such that the routing runs parallel through the datapath to avoid having area-consuming bus turns. This is especially helpful when assembling layout blocks where the protruding pins from a block (can be seen in Figure 13) snap easily into the pins of the neighboring blocks.

5.3 Buffering

Signal buffers as well as digital token buffers are added to provide better drive strengths for signals routing long distances. Signal buffers consisting of 2 strong inverters are added to the global reset signals within the channels; the reset signals,

once they enter a channel, are buffered separately in the three sub-segments of the channel to provide a fan-out structure resembling that of a synchronous clock tree. The data token stream coming in from the front end is buffered with 10 half token buffers to provide additional slack downstream. Token buffers are also added at the inputs and outputs of the arbitration circuits since these signals route long distances between channels. In critical modules such as the accumulator and NCOs, transistors are made larger and signal buffers added for more robust signal swings. It is important to note that signal buffers can only be added to non-isochronic forks. Therefore the buffers have to be added manually to the netlist and not with the automatic buffer insertion feature in commercial CAD layout tools.

5.4 Power-Grid Reinforcement

As shown in Figure 13, the layout created using cellTK and commercial CAD tools has local power and ground tracks in the horizontal and vertical directions. Metal 1 is used for the vertical local power and ground tracks and can be seen as thick blue traces in the figure. Their horizontal counterpart is routed on Metal 3 seen as thick green traces. These local power and ground signals are connected up the metal layers through vias to metal layers 7 and 8 to form a grid-like pattern as shown in Figure 16. This power grid is connected to the power ring surrounding the outer edge of the chip. The ring in turn routes external power and ground from the pads. Such power grid reinforcement is essential for even power distribution throughout the chip to prevent local brownouts.

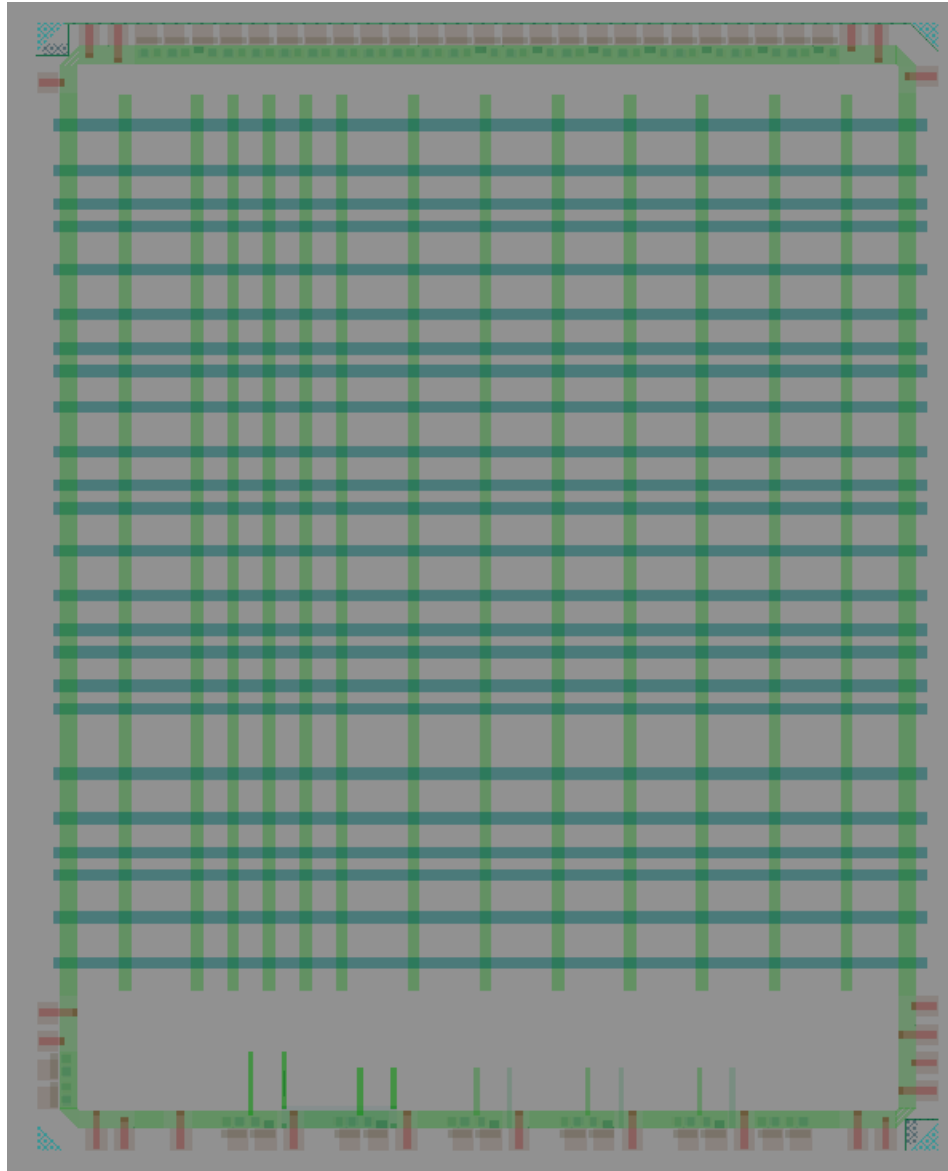


Figure 16 – Power grid reinforcement on metal layers 7 and 8

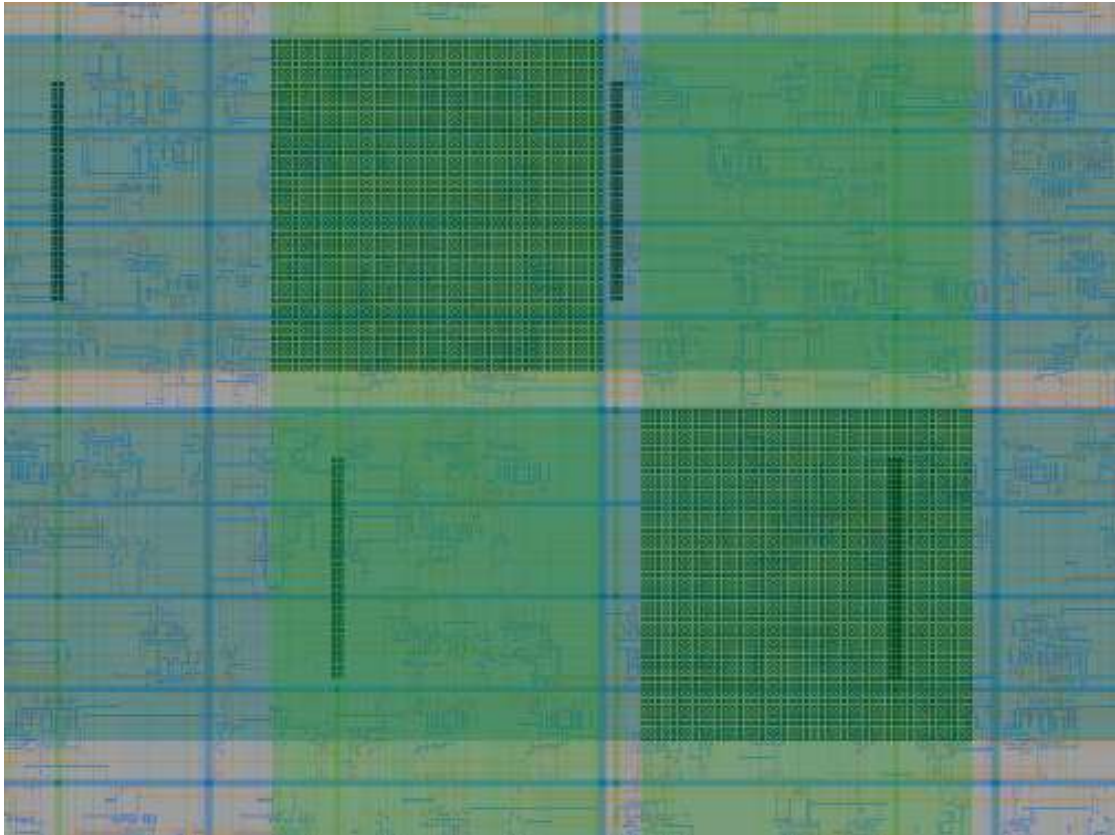


Figure 17 – Closed-up view of power grid reinforcement

In Figure 17, the thick turquoise horizontal bands are the power and ground grid on Metal 7. The thick green vertical bands are the power and ground grid on Metal 8. They are connected through a large number of vias seen as tiny square boxes in the figure. Alternating local power and ground lines can also be seen in the figure as thinner green and blue grids. They connect to the global power and ground accordingly using vertical strips of vias from Metal 3 all the way up to 7. Since they traverse through so many metal layers, these strips of vias have to be carefully placed so that they do not short out other signals. Note that for accurate power measurements, the pads and their ESD protection circuitry have a separate Vdd. The test structures also have their own Vdd and do not share the baseband processor's.

The die photo of the fabricated chip is shown in Figure 18. The pads have been wire-bonded to the package pins. The top metal layers that make up the reinforced power grid can be seen clearly. Lower layers are less visible due to the fill placed by the foundry.

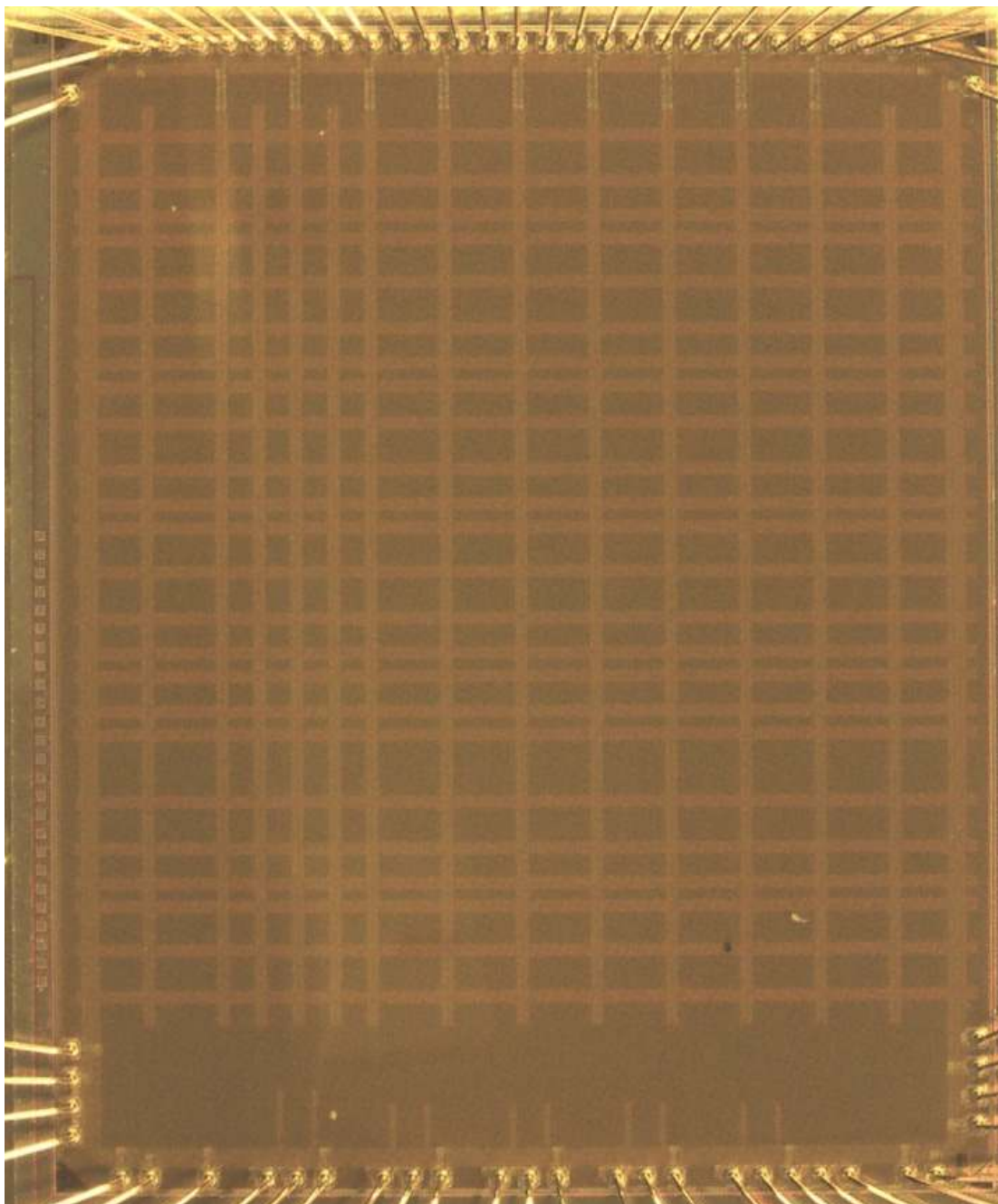


Figure 18 – Die photo

CHAPTER 6 – RESULTS

6.1 Receiver Performance Simulations

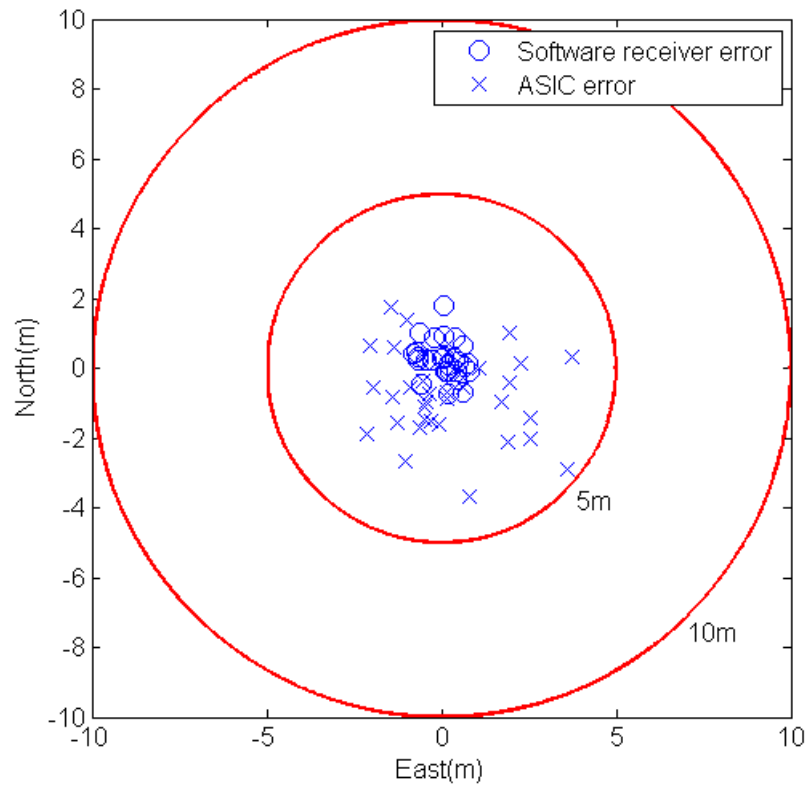


Figure 19 – Position accuracy using 6 satellites

The system is simulated with 60 seconds of satellite signals generated from the Spirent GPS simulator without atmospheric, ionospheric or multipath errors. The signal from the Spirent GPS simulator is fed into the Zarlink GP2015 front end providing digital samples with a sampling period of 175ns. These data points are recorded in a binary file that is subsequently fed into the CHP and gate-level simulations of the system. The receiver position in local coordinates is computed

from 6 satellites and compared to the actual simulated position. A comparison of position error between the system and a reference Matlab software receiver developed by the Cornell GPS Laboratory is shown in Figure 19. The inner ring is 5 meters out in the North-East local coordinates. The system 3D rms position error is 3.9m whereas the 2D rms error is 2.2m. The ASIC has a larger error spread because of the use of a single-bit RF front end samples, fixed point arithmetic and approximations.

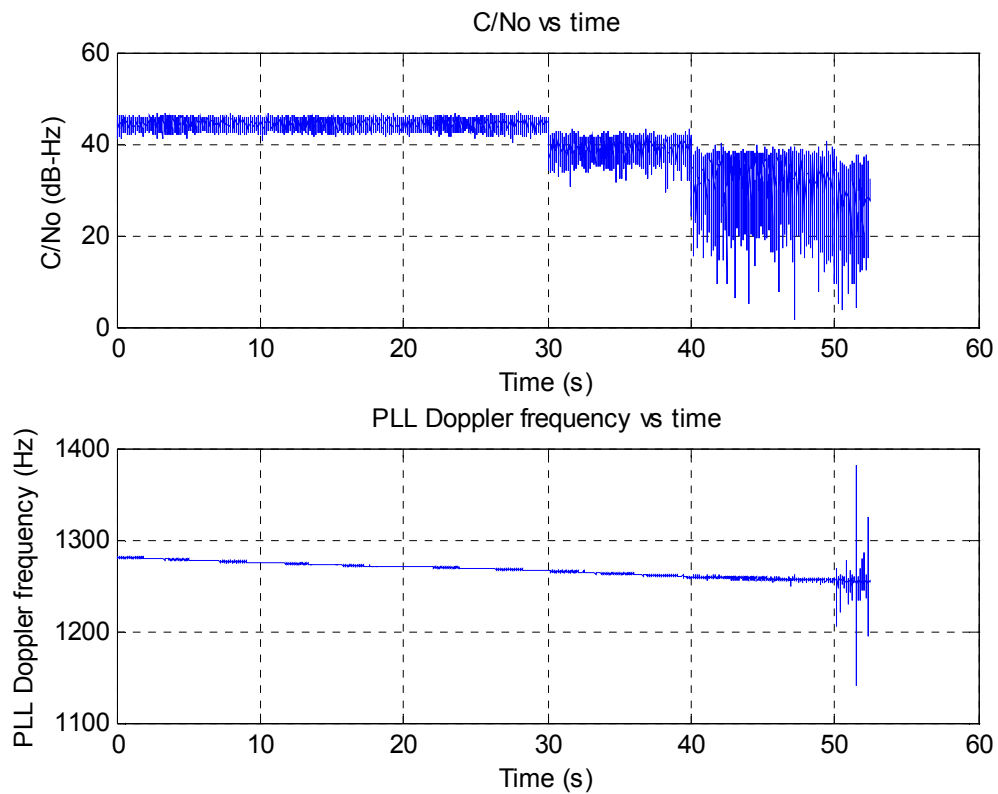


Figure 20 – Tracking sensitivity test

Tracking sensitivity is also tested with the Spirent GPS simulator using a graded reduction in carrier to noise ratio (C/No) every 10 seconds starting from the 30th second. Figure 20 shows that for C/No above 35dB-Hz, the PLL experiences no

loss of lock. Hence, the PLL performs well in normal conditions but in a weak-signal environment, longer accumulation intervals are needed to boost tracking sensitivity. If the accumulation interval is increased, width of the accumulators and the design of the bundled-data math need to be changed. As widening the accumulators corresponds to adding more bits to the IDD unit, it only marginally increases switching power. The next chapter discusses future improvements to the accumulator design. In addition, as the frequency of the bundled-data arithmetic circuits is low, the width of the accumulator can be increased without affecting power consumption significantly.

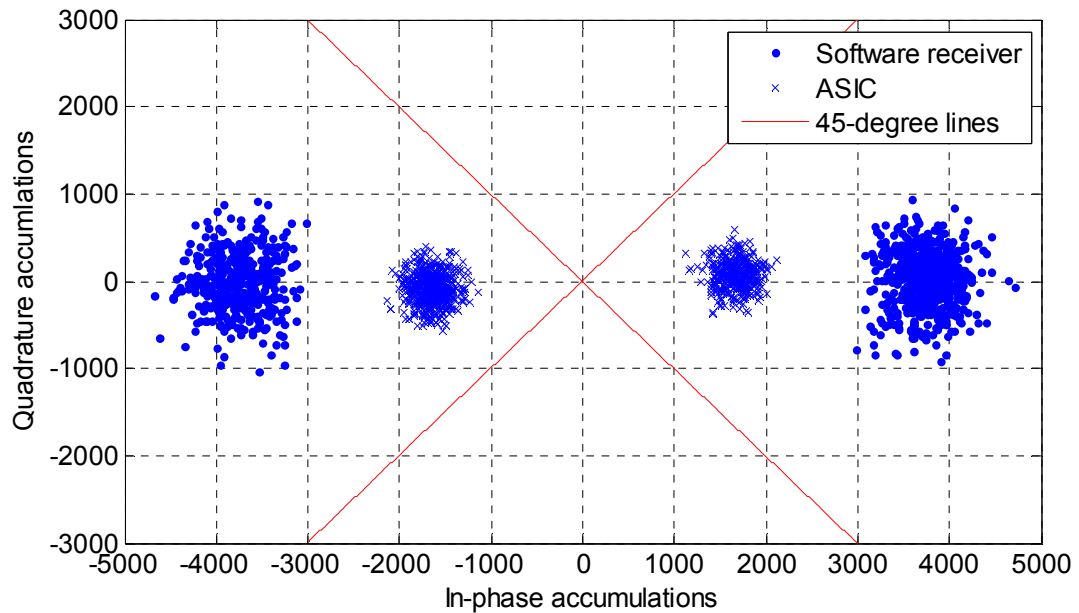


Figure 21 – In-phase and quadrature accumulations phasor comparison between the ASIC and the software receiver

The front end's ADC quantization noise also plays a part in tracking sensitivity. Figure 21 shows the distribution of the in-phase and quadrature accumulations when tracking with the PLL. The correlation power is concentrated in

the in-phase portion of the signal and the data bit flips in the signal contribute to the two blobs on each side of the vertical axis. The Matlab software receiver which uses 2-bit ADC samples has a higher correlation power than the ASIC which only uses 1 bit. Despite using fixed-point arithmetic and approximations, the ASIC's PLL tracks Doppler frequency relatively well, albeit noisier compared to the Matlab software receiver as shown in Figure 22.

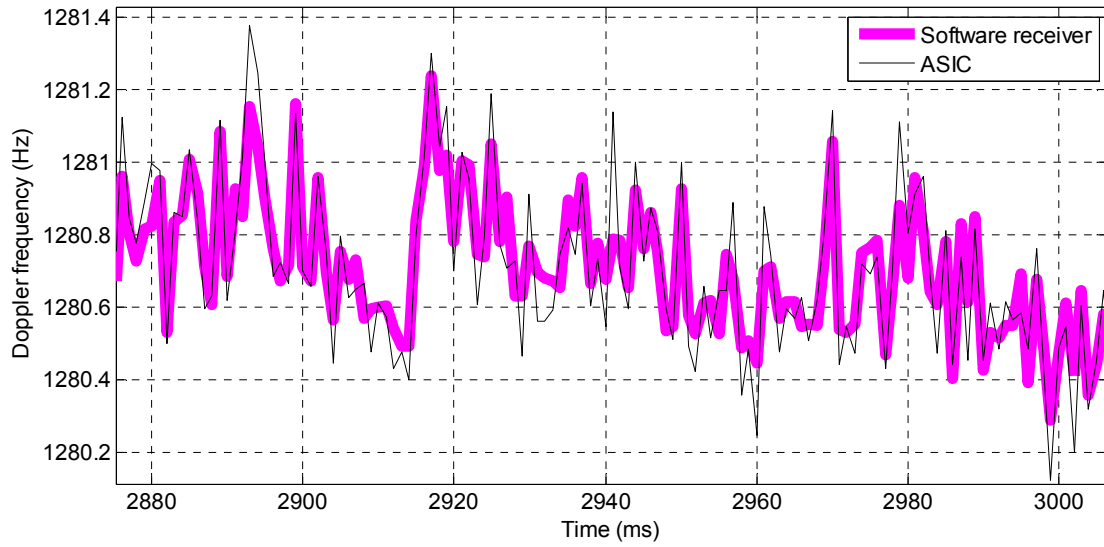


Figure 22 – PLL Doppler frequency tracking performance compared to software receiver

6.2 Power Simulations

HSIM simulations are performed on the system in a 90nm technology, at $V_{dd}=1V$ and $T=25^{\circ}C$. Wire load is added in the simulations to provide a more realistic account of the system's power numbers. The added wire capacitance is based on estimates from post-layout simulations of a small portion of the system. Table 1 shows the power consumption of various subsystems in our baseband processor for 6

receiver channels during acquisition and continuous tracking modes. Since the acquisition and tracking modes in our architecture share almost all of the fast-rate modules that make up the correlators, the power consumption in both modes are close to each other. Furthermore, because the implementation of a 6-channel system involves duplicating the channel-dependent modules in a single channel six times, the power consumption of the full 6-channel system can be derived by adding the power consumption of the arbitrated shared math modules with 6 times the power consumption of the channel-dependent modules in a single channel. The total power consumed by our 6-channel system is 1.5mW during acquisition and 1.4mW in continuous PLL tracking mode.

Subsystems		Acquisition (μW) (6 channels)	Tracking (μW) (6 channels)
Correlators	Carrier NCO	477.4	442.8
	Code NCO	439.4	400.2
	Accumulators	367.3	359.9
Code Generator		41.8	39.9
Bundled-Data		5.9	6.4
Controls, Support		158.6	163.8
Total		1.49mW	1.41mW

Table 1 – Simulated power broken down according to modules during acquisition and tracking

Name	This Work	MediaTek [6]	STMicro [14]
Process	90nm	0.11 μ m	0.18 μ m
Voltage (V)	1.0	1.2	1.6
Number of Channels	6	22	12
System Power (mW)	1.4	34.0	56.0
RF Power (mW)	-	19.5	20.0
Baseband Power (mW)	1.4	14.5	36.0
Baseband Power/Channel (mW)	0.2	0.7	3.0
Voltage-Scaled Baseband Power/Channel (mW)	0.2	0.5	1.2
3D rms Error (m)	3.9	-	3.0

Table 2 – Comparison with state-of-the-art

Table 2 provides a comparison of our system with other GPS receivers published recently. The receivers in [6] and [14] are system on chip (SOC) receivers with an integrated RF front end and digital baseband processing. To compare the per channel baseband processing power of our system with these SOC, we subtract the power consumed by their RF front ends from their system power, the result of which is then divided by the number of channels tracked. Note that these SOC have the ability to compute position estimates whereas our system stops short of doing that but provides measurements that can be used to derive position estimates in a host processor or base station. Nevertheless, the power consumed in position estimate computation is negligible for typical position update rates on the order of 1Hz. Taking all these into consideration, our per channel baseband processing power is about 3 times lower than [6] and about 12 times lower than [14]. When scaled by voltage, our per channel baseband processing power is 2.5 times lower than [6] and 6 times lower

than [14]. Our system’s position estimates, computed offline, have a 3D-rms error below 4m which is comparable to [14] and is typical for receivers used in mobile devices. The higher number of satellites tracked by [14] however does improve its position accuracy due to the effects of Dilution of Precision (DOP) [10].

6.3 Post-Silicon Testing and Measurements

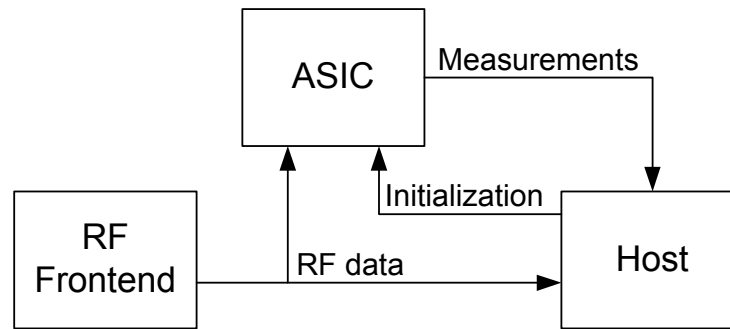


Figure 23 – Host-ASIC ecosystem with COTS frontend for testing

As shown in Figure 23, the GPS baseband processor ASIC interfaces with both an RF frontend and a host microcontroller. The host initializes the ASIC with acquisition, tracking and other programmable parameters. The ASIC then processes data tokens from the frontend through its data-flow driven architecture and outputs observable measurements to the host for position computation.

After fabrication by the foundry, the ASIC bare dice are wire-bonded and packaged in 68-pin PGA packages. I created a test setup for post-silicon testing and validation of the chip in the lab as shown in Figure 24. An FPGA is used for testing fast I/O while a microcontroller is used for chip initialization and debugging. The Cornell GPS Laboratory provides an RF frontend that runs on the Zarlink GP2015

chipset. It uses a sampling frequency of $\sim 5.714\text{MHz}$. Voltage level translators are used to interface the signal connections between the ASIC which operates at $<1.2\text{V}$ and the FPGA or microcontroller which operates at 3.3V . Keithley source meters are used to power the chip and to measure the current flow. Since the ASIC has separate Vdd pins for the pads and the system circuits, one source meter is used to power the pads and voltage translators while another is used to power the system circuit under test.

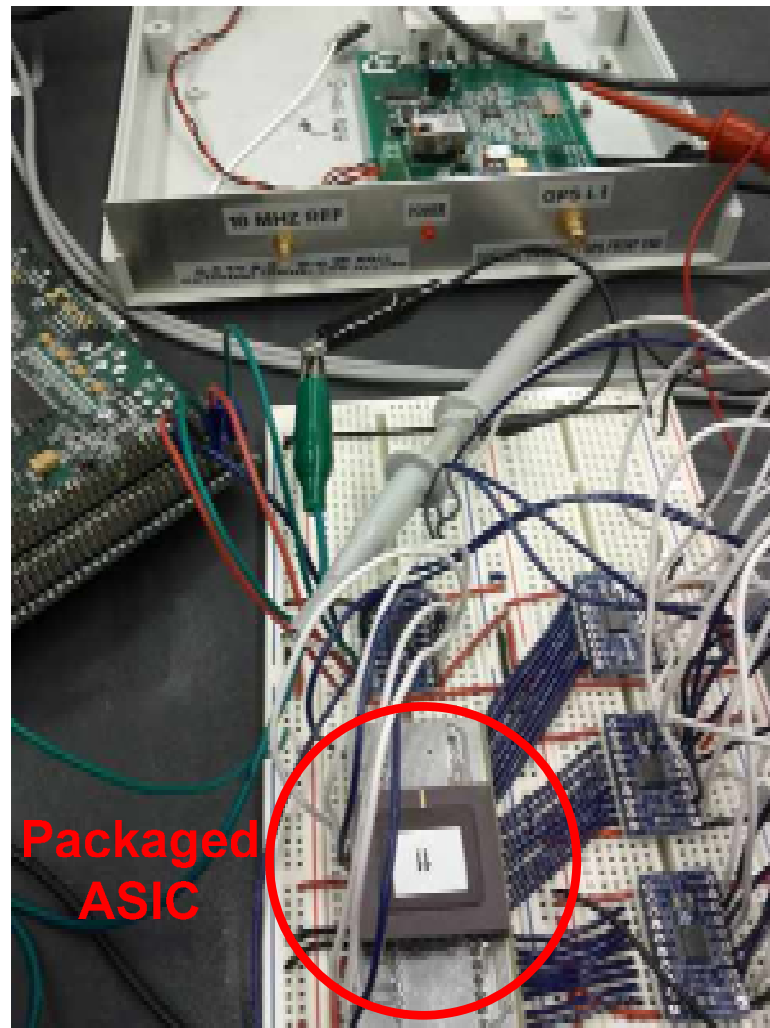


Figure 24 – Test setup for post-silicon verification

6.3.1 Test Structures

In addition to the GPS baseband processor, the ASIC chip consists of several test structures for the accumulator and Code NCO modules. The test structures allow separate testing of the key modules and analysis of their power consumption independent of the whole baseband processor system. Test structures are created from the same netlist of these key modules in the baseband processor and wrapped with test logic. The test logic for the accumulator test structure involves token sources, sinks and internal counters feeding the accumulator. The frequency of the feed is controlled by a 1-bit asynchronous channel exposed to the environment. Likewise, the Code NCO test structure also has its own 1-bit asynchronous channel that allows the user to vary its operation frequency. The carrier NCO is not included in the test structures due to its similarity to the code NCO.

The power consumption of the accumulator test structure on one of the chips is measured at different supply voltages. Though the actual rate of operation of the accumulator is the RF front end sampling frequency, measurements at different rates of operation are performed to analyze the accumulator's dynamic range. An FPGA is programmed to control the rate of operation of the test structures by completing the handshake of the 1-bit control channels at a particular frequency. Three approaches are used to obtain three key data sets on a range of different operation frequencies. The first data set involves operation at the maximum frequency. This is done by shorting the control handshake signals together. The second data set involves the actual front end sampling frequency. To emulate the test structures' integration with an actual front end, the Zarlink GP2015 front end sampling clock is used to gate the

controls from the FPGA at 5.714MHz. The third data set uses a range of realistic operation frequencies controlled by a delay counter in the FPGA.

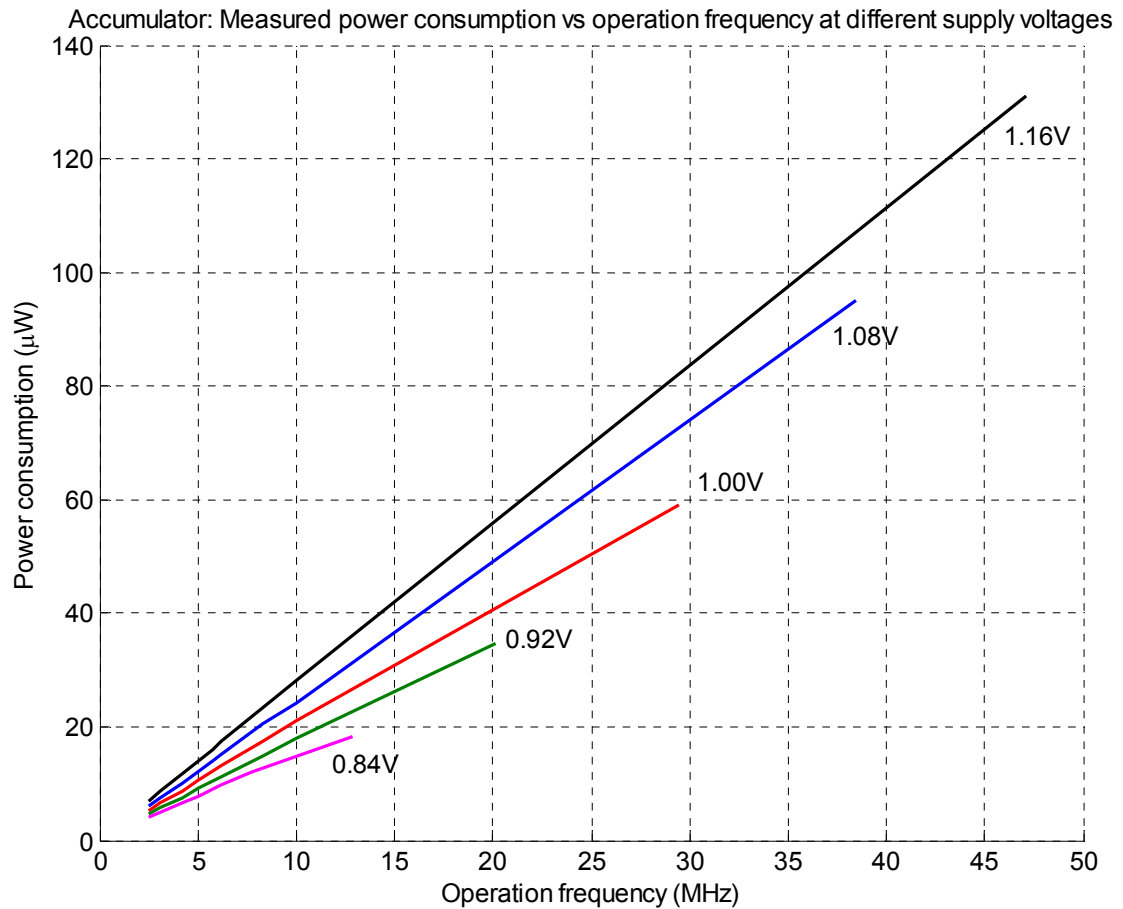


Figure 25 – Accumulator module power from measured test structure for different operation frequencies and supply voltages

Since the test structures have control and test logic in them, their power consumption must be accounted for when measuring the power of the test structures. Hence for comparison purposes with system simulation, the power measured for just the accumulator module or the NCO module is obtained as a percentage of the test structure power. These percentage breakdowns are derived from SPICE simulations

of the test structure; the accumulator module consumes 75% of its test structure power whereas the NCO module consumes 96% of its test structure power. The power numbers measured from the test structures are shown in Figure 25 for the accumulator and Figure 26 for the NCO. The power numbers for just the accumulator and the NCO modules without test logic power are derived from the percentages and are summarized in Table 3.

Figure 25 shows that the higher the voltage supply, the higher is the circuit performance. At 0.84V, the accumulator can support sampling frequencies up to 13MHz and at 1.16V, over 47MHz. Since power consumption is directly proportional to voltage squared, there is incentive to use lower voltages as long as the operation frequency is within the dynamic range. In addition, because of their inherent robustness to timing variations, asynchronous circuits can gracefully handle dynamic voltage scaling to lower power consumption. This feature is described further in the Advanced Correlators section in Chapter 7. At the Zarlink GP2015 RF front end sampling frequency of 5.714MHz, the accumulator test structure consumes 16 μ W at 1V; the accumulator module power is in turn derived from the measurement to be 11.9 μ W. As for the NCO, Figure 26 shows that it can support a maximum operation frequency of 54MHz at 1.16V. At the lower end, the NCO can run at frequencies up to 15MHz at 0.84V. The power consumed by the NCO test structure at 5.714MHz at 1V is 73 μ W; the NCO module's power is then derived to be 69.8 μ W.

To estimate the power consumed by all the accumulators and NCOs in the system, these measured power numbers are multiplied by the total number of said modules in the system. There are altogether 36 accumulators and 12 NCOs that make

up the correlators in the system. At 1V and an operation frequency of 5.714MHz, the measured power of the correlators in the system is 1.27mW as shown in Table 3. Compared to the simulated power of the accumulators and NCOs during tracking in Table 1, the measured power is 6% higher. Since the power consumed by the accumulators and NCOs make up 85% of the total system power from Table 1, the measured system power during tracking is estimated to be 1.49mW. Table 4 summarizes this comparison between simulated power and measured power.

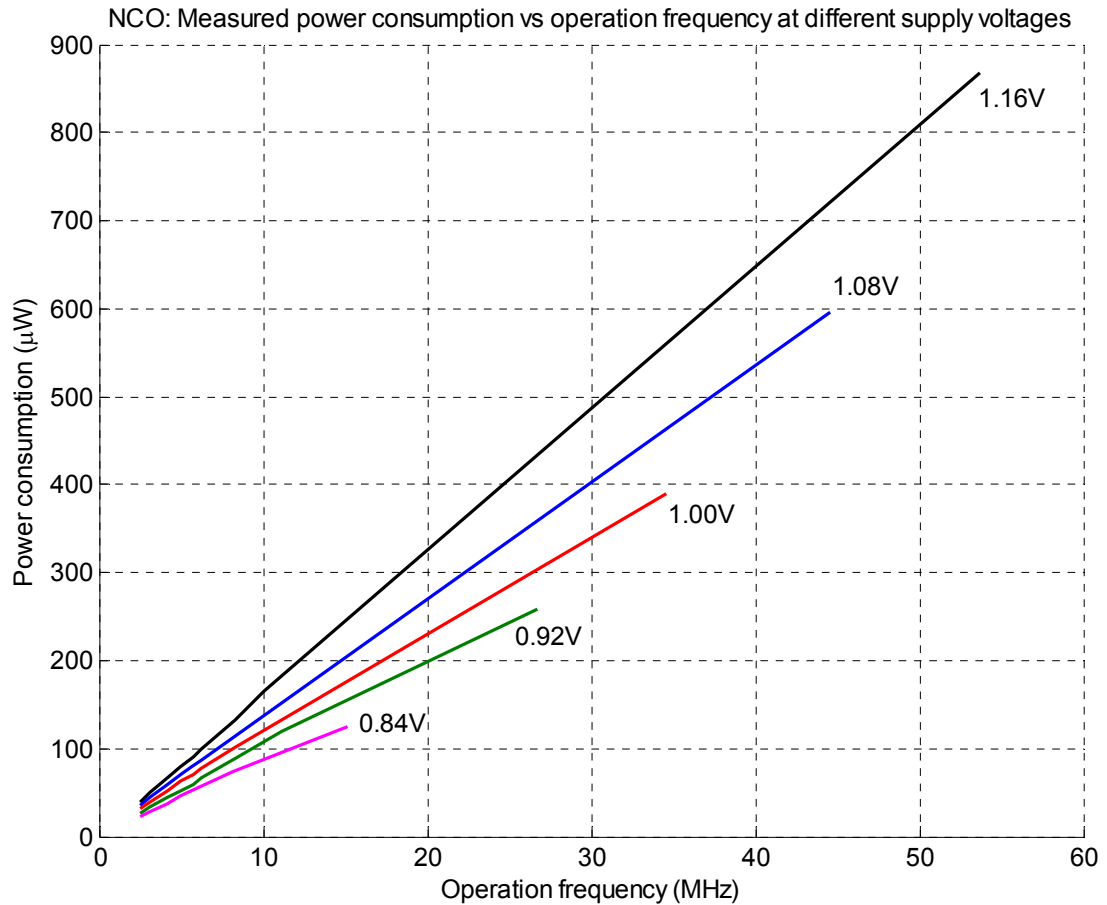


Figure 26 – NCO module power from measured test structure for different operation frequencies and supply voltages

Test Structure	Measured Power (μ W)	Module Power (Derived from Measurement) (μ W)	Number of Units in System	Power of Modules in System (mW)
Accumulator	16	11.9	36	0.43
NCO	73	69.8	12	0.84
Total Correlators Power in System				1.27mW

Table 3 – Extrapolated measured power of accumulators and NCOs for 1V, 5.714MHz

	Total Correlators Power (mW)	Total System Power (mW)
Derived from Measurement	1.27	1.49
Simulation	1.20	1.41

Table 4 – Comparison between simulated power and measured power derived from test structures

6.3.2 Full System Test

Upon power up, the full baseband processor is first initialized using a microcontroller. With the system in reset mode, the reconfigurable system parameters are loaded into system memory through a scan-chain structure. A series of readouts from memory confirms that the bits have been written correctly. Reset is then released and data tokens fed into the ASIC.

The system deadlocks after 7 data tokens; the handshake to the input stream stops responding beyond that point. Debugging signals read out indicate that the system does not make forward progress beyond the second data token and the subsequent five data tokens fill up the 10 half-buffers in the input stream. This could be consistently replicated in many different chips. However, the issue did not show up during our pre-silicon testing and verification. The parasitic capacitance is re-

extracted from the system layout and a more detailed post-layout SPICE simulation is repeated with higher accuracy. A violation to the half-cycle-timing assumption (HCTA) [19] by five wires is found.

QDI circuits are the most robust of the bunch of asynchronous logic families. Its only timing assumption is the isochronic fork assumption. While the isochronic fork assumption is more frequently discussed, the HCTA is not because it is not so much a logic problem but rather an analog circuit problem. State-holding QDI circuits require a staticizer or keeper to keep its logic driven. Figure 27 shows a C-element circuit with a weak output inverter. The output inverter is sized too small to drive a large output load, resulting in a slow output transition. If the inputs switch before the output can fully transition and if this causes the C-element to stop driving node $_Z$, the keeper will end up taking over to drive $_Z$ with the wrong logic.

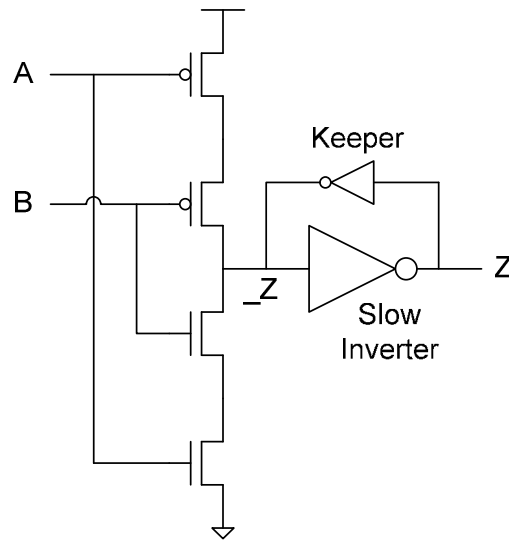


Figure 27 – C-element for half-timing-assumption illustration

In our case, the HCTA is violated by five very long wires that are driven by

weak inverters in a state-holding logic. Figure 28 shows one of the instances in the system control module where this occurs. This module passes the buffered data stream to the NCO. Referring to the figure, the signals $D.t$ and $D.f$ are input data streams from the frontend. They pass through five half-buffers to feed into this control module. The signal $B.f$ is the false rail of the buffered data stream and $B.e$ its acknowledge. $Z.f$ is the false rail of this module's output and it is a very long wire driven by a weak inverter. In Figure 29, this wire is close to 800 μm long. When $B.f$ is asserted, $Z.f$ should go high as well. However, $Z.f$ does go high initially albeit slowly. Before it can fully transition (only rises <0.1V in Figure 28), its keeper pulls it back down, resulting in a false handshake. This is caused by switching in the input of this module such that the keeper takes over to drive this state-holding logic with the wrong value from $Z.f$ that is still slowly transitioning.

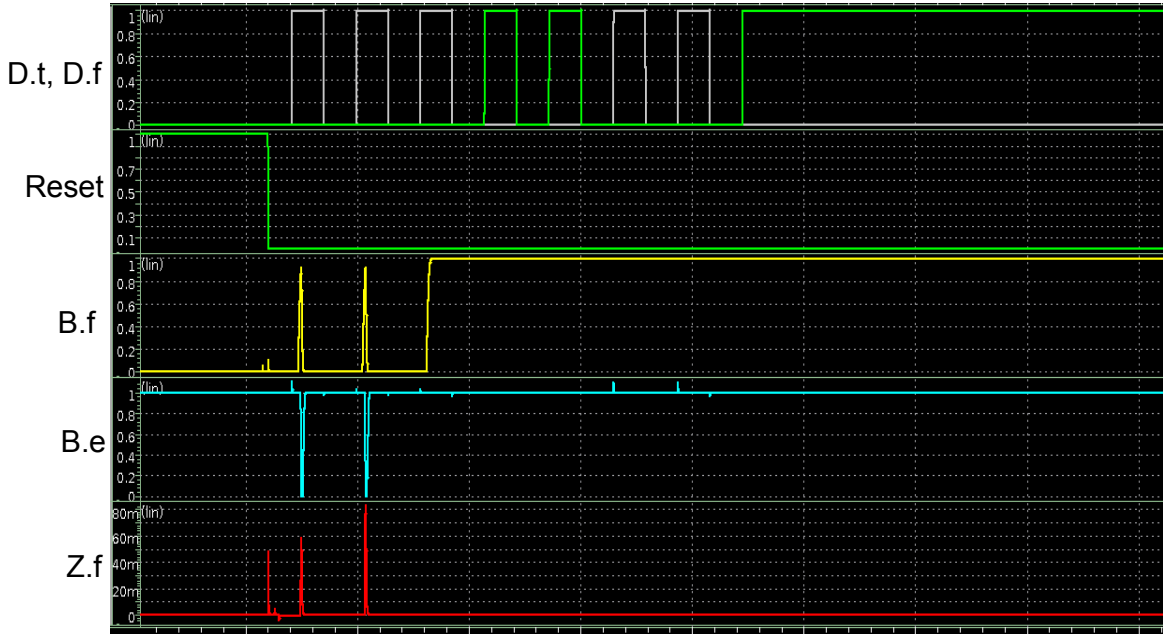


Figure 28 – Deadlock seen in detailed SPICE simulation

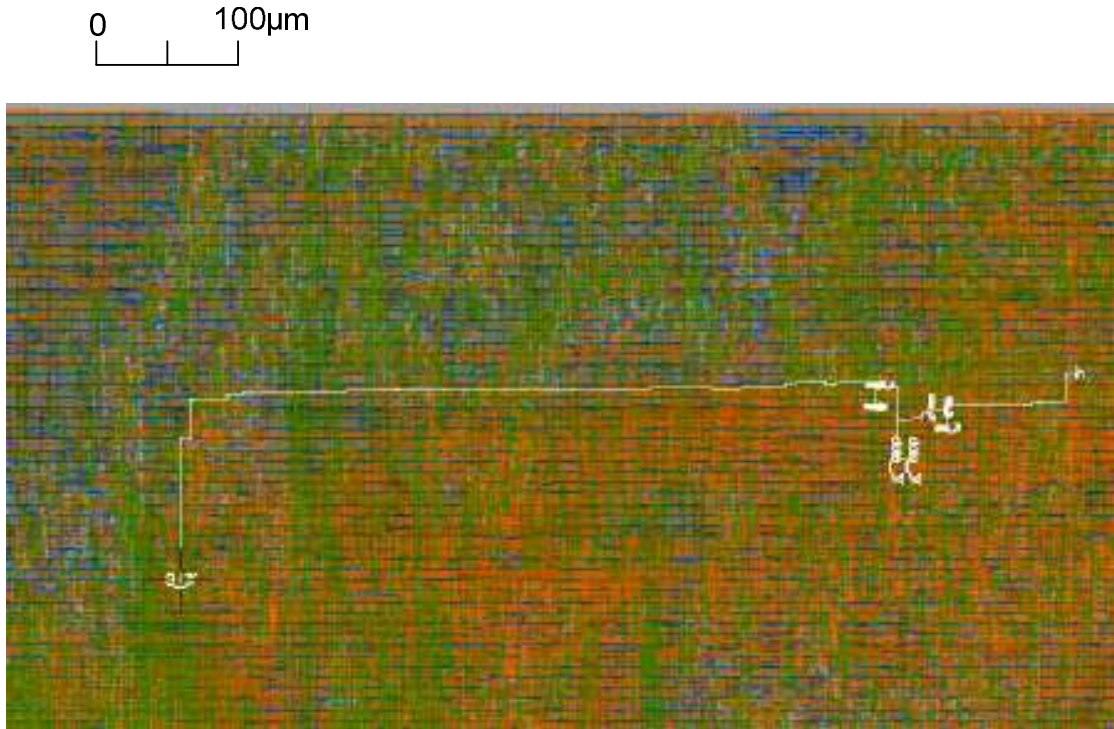


Figure 29 – Long wire driven by a weak inverter. Each segment of the scale shown is 50 microns

Similarly, the true rail of this module's output is also a long wire driven by a weak inverter. Three other similar instances are also found in the layout of the channel. The reason for such a long wire is that the layout of the channel is done in three segments using cellTK with each segment having over 100K transistors. Two supposedly neighboring modules, even in the same segment, end up being far apart in the layout.

The circuit netlist is isolated and analyzed further in Cadence. A parametric sweep of the output load and the size of the output inverter reveals that to prevent this problem, the output inverter needs to be at least four times stronger. With the fixes implemented, the deadlock problem is solved. Figure 30 shows the deadlock-free data token stream $(D.t, D.f)$ in the post-layout SPICE simulation of the revision. All the

signals including Z.f have rail-to-rail swing.

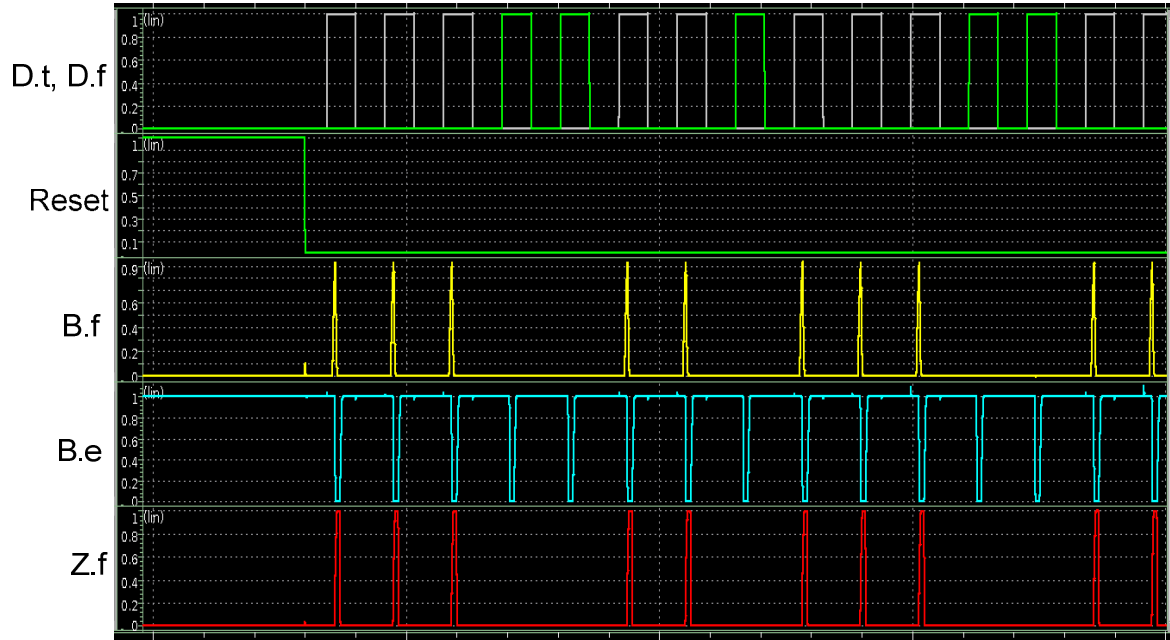


Figure 30 – Post-layout SPICE simulation of the revision without deadlock

CHAPTER 7 – FUTURE

7.1 CMOS Technology Scaling

Over the past several decades, CMOS technology scaling has been the dominant driver in improving energy efficiency of transistors. As the voltage supply is decreased, dynamic energy consumption decreases. Since the scaling of voltage supply results in reduced source-drain currents, the threshold voltage needs to be scaled accordingly as well to avoid performance penalty. However, decreasing threshold voltage inevitably increases leakage currents and we have already reached a point where CMOS technology scaling is no longer sufficient to reduce power consumption.

With the asynchronous bundled-data ASIC tracking loops design, we can achieve very low power consumption levels of about $2.5\mu\text{W}$ in the PLL and DLL. The total power consumption in the tracking loops consists of dynamic power and leakage power. Dynamic power is the power consumed by the circuit when it is actively switching. Leakage power on the other hand is caused by unwanted current passing through the transistors when they are supposedly off. With an asynchronous bundled-data design, the DLL and PLL algorithms are synthesized into their respective combinational logic blocks that stay idle when there is no computation to be done. Since there are over 200,000 transistors in each of these combinational logic blocks and that they operate at kHz frequencies, leakage power must not be overlooked. To analyze this effect of scaling, I used a commercial synthesis tool and standard cell libraries in 40nm, 65nm and 90nm technologies to synthesize the

tracking loops algorithms for different number fractional bits. Note that these standard cells are provided by a different foundry than that which fabricated our ASIC. The power consumption of the tracking loops circuits was also analyzed for different operating frequencies determined by the number of channels sharing the tracking loops. The effective operating frequencies of the tracking loops is 2 kHz for a single channel with 1ms accumulation period and 2N kHz for N channels.

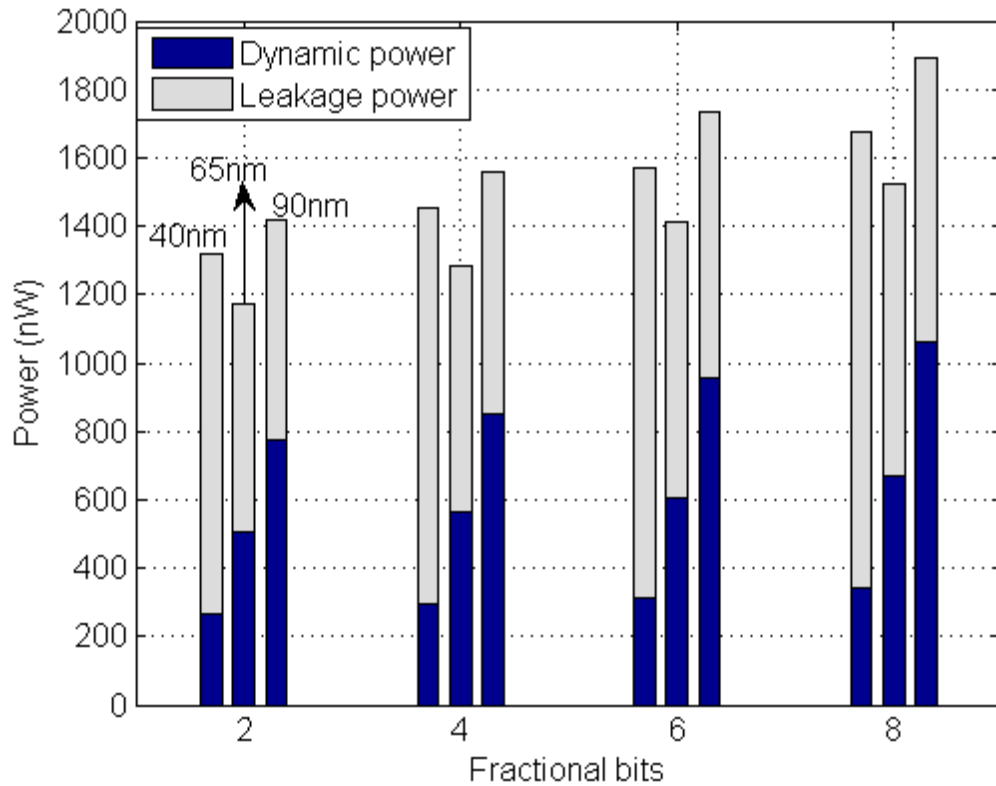


Figure 31 – Effect of the number of fractional bits on leakage and dynamic power of the tracking loops shared by 5 channels, for 40nm, 65nm and 90nm technology nodes

Figure 31 shows the change in dynamic power consumption and leakage power as the number of fractional bits is increased. These power numbers are obtained by

synthesizing the tracking loops algorithm for five channels. The power consumed by the arbiters and muxing and demuxing circuits are negligible and are omitted from this analysis. As the number of fractional bits increases, the circuit complexity increases and hence the total power consumption of the tracking loops increases. An increase in one fractional bit contributes to about 5% increase in dynamic power as well as leakage power. As technology scales, dynamic power improves but leakage worsens as transistors' feature size gets smaller. For five channels sharing the same tracking loops, the 65nm technology is better than 40nm technology where leakage power dominates.

If there are more channels sharing the tracking loops, the tracking loops are more active and the dynamic power increases. Since dynamic power improves as technology scales, Figure 32 shows that the dynamic power consumption in the 40nm technology only increases marginally with the increase in the number of channels. For less than 8 channels sharing the tracking loops, the tracking loops are not as active and leakage power dominates. In this case, the 65nm technology is better than the 40nm technology in terms of total power consumption. However, if there are more than 8 channels sharing the tracking loops, the tracking loops are active enough that the 40nm technology overtakes 65nm as the technology of choice.

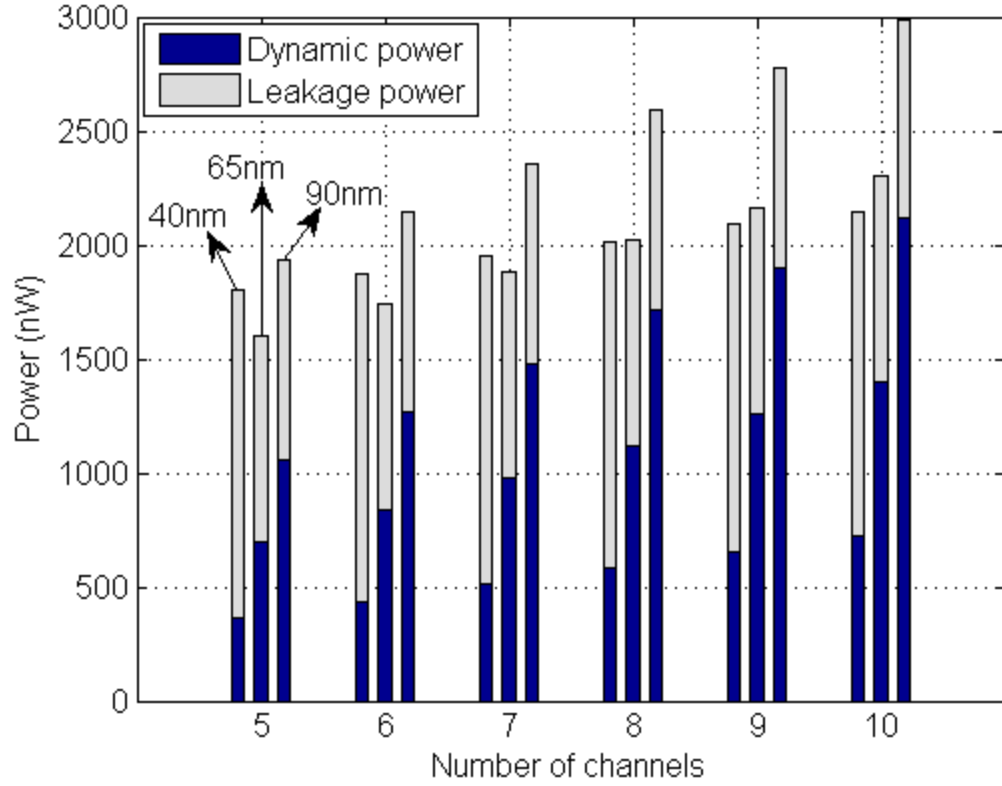


Figure 32 – Effect of the number of channels on the power of the tracking loops with 8 fractional bits, for 40nm, 65nm and 90nm technology nodes

As for the rest of the system which is QDI, many transistors are idle most of the time. Due to the architecture's data-flow driven design, only the fast modules interfacing with the frontend operate at the sampling frequency whereas the rest of the subsystems down the datapath operate at much slower rates. In addition, even within a fast subsystem, only some portions of the circuit operate at high frequencies because QDI allows us to implement data-dependant switching at the bit-level to reduce transistor activity. An example is the system's counter-based accumulator design. Hence it is noteworthy that asynchronous circuit design helps with dynamic power consumption but as transistor leakage worsens in 40nm technology and smaller, static power from the idle transistors start to be as high as the dynamic power. Power gating

of asynchronous circuits to alleviate this leakage problem have been explored in [20]. Another option is to use new technologies in asynchronous designs. MEMS/NEMS relays have many good properties that make them a promising candidate to replace CMOS in asynchronous designs. The potential of asynchronous MEMS/NEMS technology is discussed in the next section.

7.2 MEMS/NEMS

CMOS technology scaling has reached a point where the static power in transistors is as high as the dynamic power. While further process scaling will only worsen leakage in transistors, it will benefit NEM relay technology. As asynchronous circuit design helps with dynamic power and NEM relays with static power, the use of NEM relays in asynchronous VLSI is ideal for low-power applications.

7.2.1 NEM Relay Basics

NEM relays have been shown to be a very energy-efficient alternative to CMOS. [21] demonstrates the use of four-terminal NEM relays in common VLSI building blocks such as logic gates, adders, and latches. In simulation, a 32-bit adder implemented using NEM relays has been shown to achieve 10X lower energy per operation at 0.5GOP compared to its optimized CMOS counterpart. While [21] uses relays with folded flexures, [22] uses cantilever structures and has also shown an order of magnitude improvement in adders, ADC and DAC designs compared to CMOS. Other variants of NEM relays have also been proposed. For example, [26] shows a

laterally-actuated relay design and [27] illustrates the use of six-terminal relays to allow compact implementation of complex logic. This section builds upon the findings in [21] to create novel logic topologies that are effectively implemented using normally-open and normally-closed NEM relays. These topologies leverage the strengths of NEM relays to further enhance asynchronous circuit designs.

A NEM relay in the normally-open configuration is typically a 4-terminal switch that consists of gate, drain, source and body electrodes. The floating gate is suspended over the drain, source and body terminals through a spring in the form of a cantilever or folded flexures as shown in Figure 33. The relay utilizes electrostatic force to turn it on or off. When the applied voltage between the gate and the body (V_{gb}) reaches the threshold voltage also known as the pull-in voltage, the electrostatic force generated is strong enough to pull the floating gate structure towards the body, connecting the source and drain. The relay is now switched on. To turn the relay off, V_{gb} is reduced so that the gate pulls away from the body. This happens when the spring's restoring force overcomes the decreasing electrostatic force. Another type of NEM relay is the normally-closed configuration. It uses the same basic electrostatic principle as the normally-open configuration except that the switch's default state is on. When V_{gb} exceeds the threshold voltage, the source and drain terminals are disconnected instead; the switch turns off.

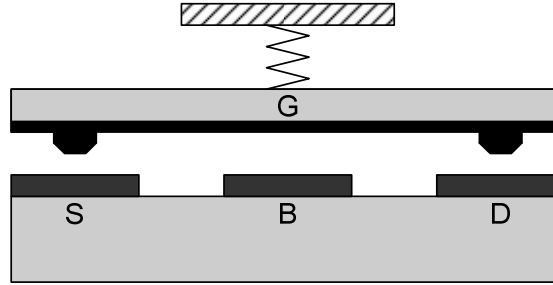


Figure 33 – Cross section view of a normally-open NEM relay

7.2.2 Case for Asynchronous NEMS

While asynchronous circuits enjoy many benefits due to its self-timed properties, their implementation in CMOS has limitations and introduces several weaknesses. Due to leakage through the transistors, all circuits, including those involved in handshakes and logic, consume static power when they are not actively switching. Besides, all logic gates have to remain driven as leakage can destroy their states over time; state-holding logic that is common in QDI circuits employs staticizers. Leakage will only get worse with further transistor technology scaling.

Secondly, pull-up and pull-down stacks must be kept small; there must not be too many series transistors in the stacks. Otherwise, the output will have a poor rise or fall time and the short circuit currents through the pull-up and pull-down stacks, which are both momentarily on, will be large. As a result, for circuits with a large fan-in, a tree structure is used. For example, completion trees are common in QDI circuits handling larger number of inputs or outputs.

Thirdly, production rules for QDI circuits have to be bubble-reshuffled. Bubble-reshuffling is a procedure that converts variables in the production rules to their correct inverted and non-inverted senses for them to be CMOS-implementable.

This procedure has to be done because PMOS and NMOS do not have similar pull-up and pull-down capabilities; the weak pull-down nature of PMOS requires that all variables involved in the pull-up network be inverted. Conversely, the weak pull-up nature of NMOS necessitates that all variables used in the pull-down network be uninverted. Hence, bubble-reshuffling in CMOS adds circuit complexity and transistor count.

Due to the physical nature in which NEM relays switch, they have several advantages over CMOS. First and foremost, NEM relays behave like perfect switches with zero leakage. They turn on or off abruptly when V_{gb} crosses the threshold voltage. There is no sub-threshold current flowing between the source and drain because the physical conduction path between them is broken when the gate switches. Hence NEM relays are useful in eliminating static power consumption, especially in asynchronous circuits where we can then truly claim that power is only consumed during useful computation. In contrast, leakage in CMOS is increasingly worse as technology scales. In fact, leakage power in modern CMOS technologies below 65nm is so dominant over dynamic power consumption that asynchronous circuits need to address it with power-gating techniques [20].

Secondly, NEM relays have sharp voltage-transfer-characteristic (VTC) curves. Unlike CMOS, the steep rising and falling edges for NEM relays contribute to minimal short circuit currents during switching. More NEM relays can be stacked together in series in longer pull-up or pull-down network compared to CMOS transistors.

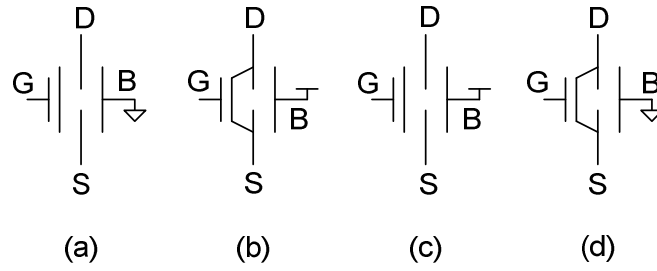


Figure 34 – (a) “NMOS” with normally-open relay (b) “NMOS” with normally-closed relay (c) “PMOS” with normally-open relay (d) “PMOS” with normally-closed relay

Thirdly, NEM relays are ambipolar by nature. Since electrostatic force is independent of the polarity of V_{gb} , as long as the voltage difference between the gate and body is larger than the threshold voltage, the relay switches. Hence, the same relay can be configured to behave like a PMOS or NMOS by connecting the body to Vdd or ground as shown in Figure 34. This also gives NEM relays both a strong pull-up capability as well as pull-down. Figure 34(c) shows a static-CMOS-like implementation of a NAND gate. Because of the weak pull-down nature of PMOS and weak pull-up nature of NMOS, an AND gate has to be implemented from a NAND coupled with an inverter. NEM relays on the other hand can implement an AND gate directly without an inverter as shown in Figure 34(d). A buffer can also be implemented directly in Figure 34(b) without the need for two inverters. Although normally-open NEM relays are shown, these gates can similarly be implemented with normally-closed relays by swapping Vdd and ground at the body terminals. The ability of NEM relays to implement both inverting and non-inverting logic is especially useful for QDI asynchronous circuits where the logic can be synthesized directly without bubble-reshuffling. Moreover, staticizers for state-holding logic that are common in QDI circuits can be implemented with just a NEM relay buffer.

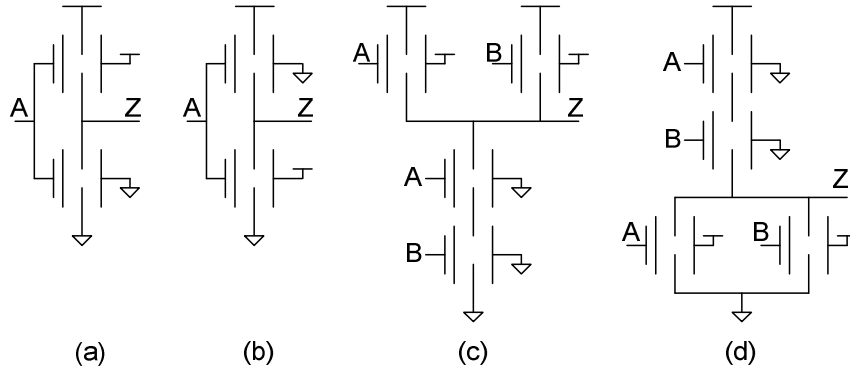


Figure 35 – Normally-open NEM relay implementation of (a) Inverter, (b) Buffer, (c) NAND, and (d) AND

On top of that, NEM relays give circuit designers plenty of flexibility to reduce device count. Pass transistor logic implemented with NEM relays is especially attractive. Not only is device count reduced, the relays can pass both a strong ‘1’ and ‘0’. Research has shown that CMOS logic can be remapped using NEM relays in pass-gates topology to optimize the delay through the circuit [23]. Besides, some logic gates can be implemented very effectively using NEM relays. Figure 36 (a) and (b) shows that the AND and OR gates respectively can be built from just two relays. The V_{gb}-induced electrostatic switching principle in NEM relays is also ideal for XOR and XNOR gates. They can be implemented with just a normally-open relay and a normally-closed relay as shown in Figure 36 (c) and (d). To further reduce device count, these combinational logic gates can be built using ratioed logic topologies. For example, an XOR gate can be implemented with just a normally-open relay and a resistive load that replaces the pull-down relay in the XOR in Figure 36 (c). The resistive load however must have resistance much greater than the on-resistance of the relay for the output to have fuller voltage swing. The trade-offs for using the ratioed logic topology of course include static power dissipation and lower voltage swing.

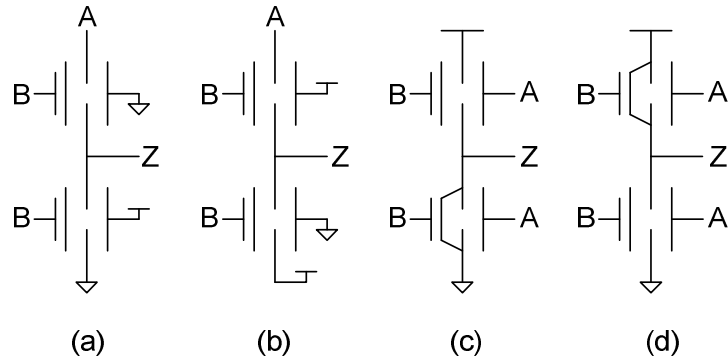


Figure 36 – Effective NEM relay implementations of (a) AND, (b) OR, (c) XOR, and (d) XNOR

In short, NEM relays have many advantages over CMOS and are very attractive for asynchronous circuit design. They not only consume zero static power and have sharp rising and falling edges, but also can implement inverting and non-inverting logic and full-swing pass gate topologies effectively with minimal number of relays. However, anything that has mechanical moving parts has limited lifetime. NEM relays' electrodes are subjected to wear and tear during switching but studies have shown that the relays are still functional over 60 billion cycles [21]. Besides that, NEM relays are slow compared to CMOS. Their slowness comes not from electrical delay but mechanical delay. Electrical delay from junction and parasitic capacitance in a NEM relay is negligible, in the order of picoseconds, and thus contributes to its sharp VTC curve. Mechanical switching delay in contrast is in the order of 10's of nanoseconds in a 90nm technology [21]. The art of balancing mechanical and electrical delays is key to exploiting NEM relays for high performance VLSI applications. Large complex logic is remapped into circuit topologies that use less relays and facilitate simultaneous mechanical switching [23]. This technique not only favors synchronous circuit modules in the critical path but also the bundled-data

design in asynchronous circuits. For mobile applications that emphasize low-power operation rather than high performance, more distributed, parallelized execution of logic functions without a global clock constraint is preferred. This allows the circuit speed to change dynamically to achieve data-flow driven, average-case performance that is one of the salient qualities of asynchronous QDI circuits.

7.2.3 QDI Synthesis with NEM Relays

Numerous NEM relay configurations afford us with circuit design flexibility. First of all, we can choose to use normally-closed or normally-open NEM relays. Secondly, the relays can be configured to be active high or low by connecting its body terminal to Vdd or ground. Like in CMOS QDI, production rules are derived from handshaking expansions. These production rules form the pull-up and pull-down network to drive a signal. However, unlike CMOS, the handshaking expansions do not have to be bubble-reshuffled for NEM relays since they can implement non-inverting logic directly. For example, referring to the NOR gate in Figure 39, to turn on a normally-open NEM relay when a variable is asserted, the body terminal of the relay is grounded. As such when the variable is asserted, there exists a voltage difference between the gate and body terminals to electrostatically actuate the relay. This is seen in the pull-down network of the L.e signal. Conversely, if a relay is to be turned on when a variable is de-asserted, the body terminal of the relay is connected to Vdd. This is seen in the pull-up network of L.e.

Therefore, non-bubble-reshuffled production rules can be implemented efficiently using NEM relays in the static CMOS-style. Existing QDI synthesis tools

can be easily modified to accommodate NEM relays. Implementation of QDI logic with pass gates style is also possible but requires additional thought and analysis for synthesizing complicated logic.

One of the weaknesses of QDI asynchronous circuits is completion overhead. The completion latency grows with the datapath width of N bits, and is proportional to $\log N$ for tree structures. One way of reducing this overhead is by using single-track communication protocols seen in [23], [24], [25]. Unfortunately, some timing assumptions and careful timing analysis need to be made for these types of circuits. Another way of reducing completion overhead while still preserving timing robustness of QDI is through the use of pipelined completion [26]. This is done with more localized completion detection at the level of a single or several bits rather than a giant completion tree for all bits. Consequently, the completion detection is no longer on the critical path. While the completion latency can be hidden behind actual logic computation delay, using detecting completion cannot be avoided with the standard OR gates and C-element. Therefore, an efficient method for implementing these gates is still critical.

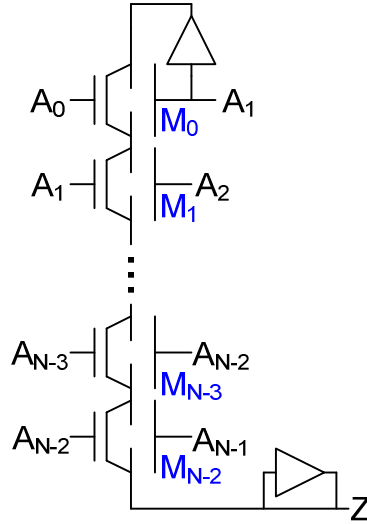


Figure 37 – N -input C-element implemented with NEM relays

Figure 37 shows how the C-element can be implemented effectively using NEM relays. A buffer is used as a staticizer for the output. Another buffer is used to avoid having a self-driving gate by one of the inputs so that the relay switching delay is taken into account. Overlapping pairs of inputs are compared against each other and whenever any input differs from its pair, the normally-closed relays turn off, allowing the staticizer to hold its output state. Since NEM relays are used here as full-swing pass gates, we can daisy-chain many of them in series to accommodate completion detection of a wide datapath. Even for a 2-input C-element, the NEM relay implementation only uses almost half as many devices compared to CMOS. Moreover, for every additional input, we only need one additional relay whereas CMOS needs two. Not only that, there is a limit to the number of transistors stacked in series for CMOS; to avoid slow circuits, large slew rates and large short circuit currents during switching, the number of transistors in series in the pull-up or pull-down network is usually kept to four or less. As a result, C-elements implemented in

CMOS have to resort to using a tree structure. The number of relays needed for an N -input C-element is $(N-1)+4$ where 4 is the constant overhead due to the relays in the drive buffer and staticizer. As for CMOS, using 2-input C-elements arranged in a tree structure, the number of transistors increase much faster with respect to the number of inputs, to the tune of $8N$. For a 64-bit data path, the C-element implementation with NEM relays takes only 67 devices compared to a whopping 512 transistors in CMOS! Even with a more optimized tree structure comprising 2-input and 4-input C-elements, 252 transistors are still needed for 64 bits.

The energy per operation and the delay of the C-elements for different number of inputs is analyzed. Simulations of the NEM relays circuit topologies in this dissertation are done in Cadence using a Verilog-A model for the NEM relays. The model parameters are scaled for the 90nm technology node and are provided in [21]. Normally-closed NEM relays are also simulated with the same model parameters except with the opposite turn on/off condition at the threshold voltage. A 25fF load is used to simulate wire and loading from other NEM relays or CMOS transistors. The NEM relay version uses the topology shown in Figure 37, and minimum sized. The CMOS version uses a tree topology of cascaded 2-input C-elements, have average transistor size of 2.5X the minimum gate length, and are SPICE-simulated in a 90nm technology. Figure 38 compares the energy per operation and delay of the C-elements with the number of inputs ranging from 2 to 64. As the number of inputs increases, capacitive loading and on-resistance from NEM relays stacked together in series increase. Consequently, the electrical RC delay increases fairly linearly with the number of inputs. However, the NEM relay mechanical delay at $\sim 20\text{ns}$ dominates; in fact, even for a 64-input C-element, the electrical delay is still about one-eighth of the

mechanical delay. Hence the method of having long stacks of NEM relays is preferred so that only a single mechanical delay is incurred. In contrast, CMOS is over 4 times faster than NEM relays for fewer bits but about 2 times faster for 64 bits. This is not only helped by faster switching in transistors but also the logarithmic increase in the number of stages and latency in the CMOS C-element tree structure, with respect to the number of inputs. Though NEM relays lose out in terms of latency, they excel in energy per operation. For a 64-bit C-element with operation frequency of 10MHz, NEM relays have over 16X lower energy per operation compared to CMOS. This is because NEM relays have zero leakage and minimal short circuit currents during switching. Furthermore, much fewer relays are needed to build C-elements, almost an order of magnitude fewer for 64 bits compared to CMOS as discussed earlier.

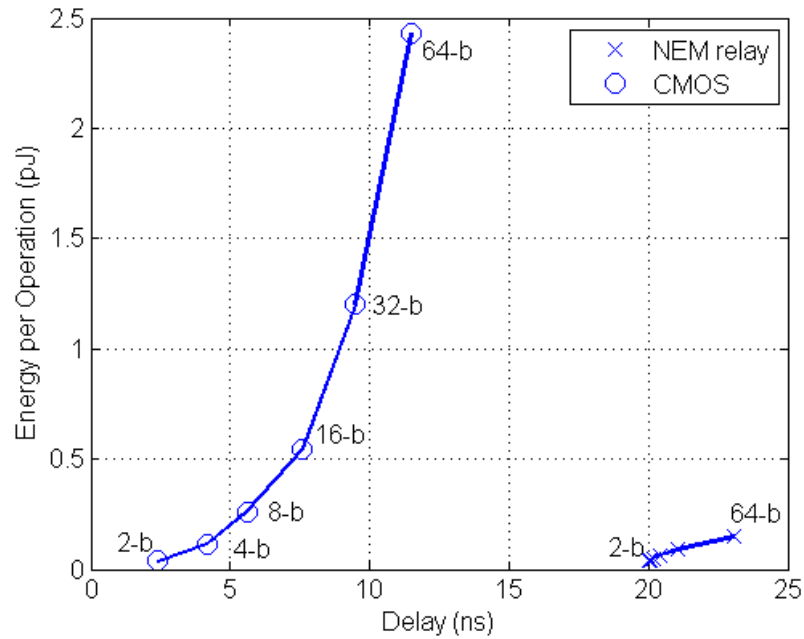


Figure 38 – Energy per operation versus delay comparison for C-element with different number input bits, implemented with NEM relays and CMOS to drive a 25fF load

7.2.4 QDI Templates using NEM Relays

The Weak-Conditioned Half Buffer (WCHB) template is fast and is popular for implementing buffers. In CMOS, the WCHB has a forward latency of 2 transitions and a cycle time of 10 transitions. On the other hand, WCHB implemented with NEM relays have a forward latency of 1 transition and a cycle time of 4 transitions. The ability of NEM relays to implement non-inverting logic directly allows it to achieve the minimal possible number of transitions. Figure 39 shows a NEM relay implementation of the WCHB with 1-bit input (L) and 1-bit output (R).

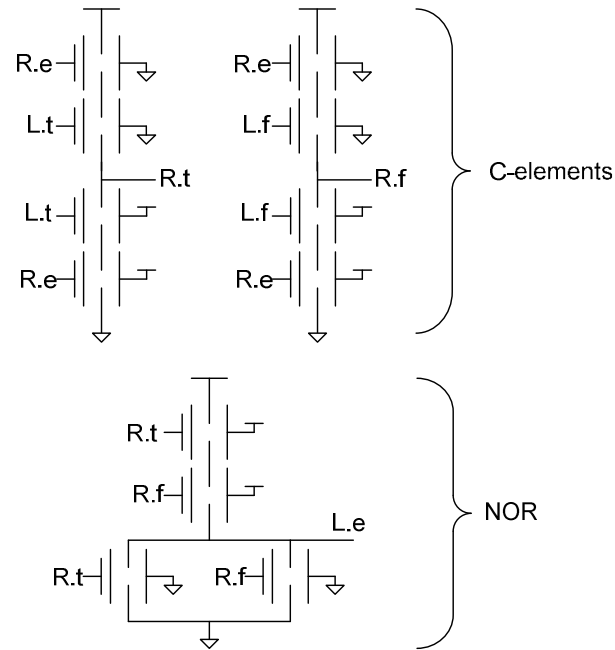


Figure 39 – WCHB with 1 bit input (L) and output (R) using NEM relays, staticizers omitted

The Pre-Charge Half Buffer (PCHB) is another popular QDI template. It is commonly used for buffered logic. With CMOS implementation, the PCHB has a forward latency of 2 transitions and a cycle time of 14 transitions. In contrast, the NEM relay version, again, achieves the minimal possible number of transitions, with a

forward latency of 1 transition and a cycle time of 4 transitions. Figure 40 shows a NEM relay example of the PCHB with 2-bit inputs (L0, L1) and a 1-bit output (R). Logic can be easily packed into the pull-up stacks of the output data rails. Besides that, the input and output completion detection circuit essentially generates the input acknowledge signal (L.e). For a large number of inputs and outputs, such as in buffered merge, split or PCHB copy modules, the pull-up stack for the input acknowledge (L.e) can become quite long. In CMOS implementation, these long transistor stacks need to be broken down into stages, resulting in increased number of gate transitions and transistor count. NEM relays however do not face such limitations; stacks can be long and still optimal providing the electrical delay through the stack is smaller than the mechanical delay. In fact, as observed in simulations in a 90nm technology, stacks can be as long as over 200 relays in series. To extend beyond this number, one approach is to reduce device count with the novel C-element topology introduced earlier. This adds an extra transition to the overall cycle time for dataless channels or 2 transitions for dualrail channels since OR gates are needed too.

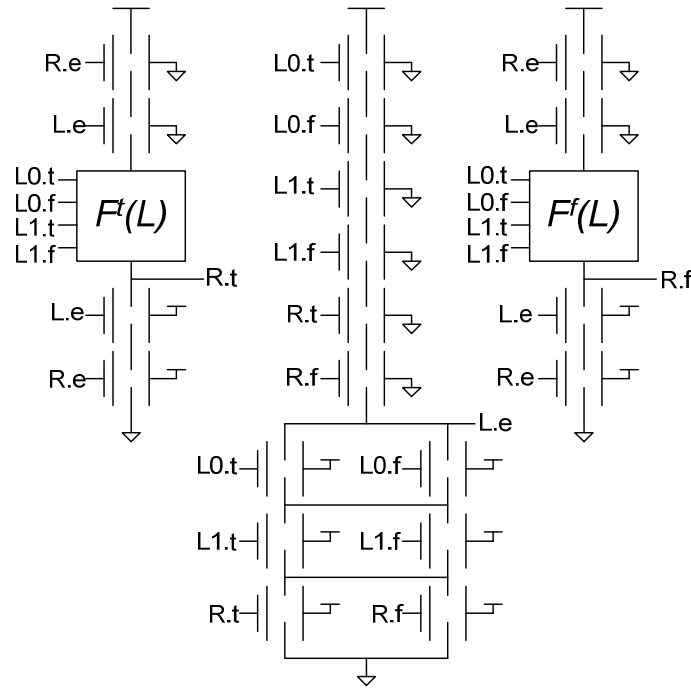


Figure 40 – PCHB 2-bit input and 1-bit output using NEM relays, staticizers omitted

It is important to note that although QDI templates implemented with NEM relays exercise much fewer transitions in a cycle compared to CMOS, each transition in current NEM relay technology may be an order of magnitude slower than CMOS due to mechanical delays. This can be seen in Figure 41 taken from a scaled 90nm NEM relay technology simulation of the PCHB with dataless input (L) and output (R) channels. The good news is that further relay technology scaling will improve mechanical delays. Besides that, the same NEM relay synthesis and design techniques introduced here are also applicable to other QDI templates to give us the smallest possible number of transitions in a cycle.

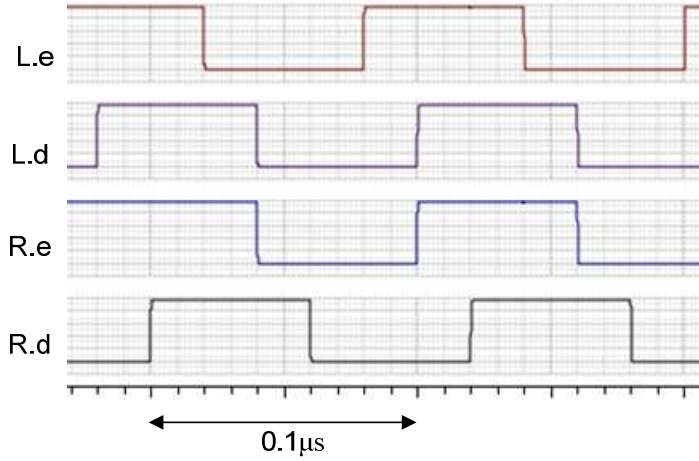


Figure 41 – Simulation of the PCHB dataless channel with NEM relays

AND is used as an example of simple logic functions that can easily be incorporated into the PCHB template. For bitwise AND function, as the number of bits increases, the output data rails' circuit structure remain the same but the input acknowledge's (L.e) circuit as shown in Figure 40 becomes more complex. The number of transistors in series in the pull-up stack of L.e will be $6N$ and its pull-down stack $3N$, where N is the number of bits. As mentioned earlier, CMOS implementation will need a completion tree to break down such long stacks whereas NEM relays do not need to. NEM relays can implement long pull-up and pull-down stacks, which not only reduces device count, but also power and cycle time.

Implementing an N -input AND function is slightly more complicated but allows emphasis of the flexibility of NEM relays over CMOS designs. For a 2 bit AND function, the logic function in the output true rail will be $L0.t \wedge L1.t$ whereas that of the output false rail will be $(L0.f \wedge L1.f) \vee (L0.f \wedge L1.t) \vee (L0.t \wedge L1.f)$. As the number of bits increases, the logic function in the output false rail increases exponentially. Hence, for a large number of inputs, the AND function is usually

implemented by cascading smaller function blocks in a tree structure. With the NEM relays, the output false rail logic function can be greatly simplified. The validity of the input rails can be checked directly by using the body and gate terminals of the NEM relays as shown in Figure 42.

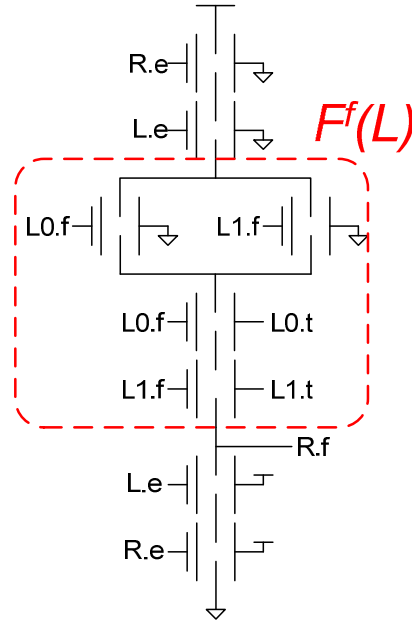


Figure 42 – Optimized output false rail of a 2-input PCHB AND using NEM relays

The PCHB AND function is simulated with different number of inputs for CMOS and NEM relays. All signals interfacing with the environment are buffered accordingly with transistors or NEM relays. To accommodate the slowness of NEM relays implementing four-phase handshakes, the circuit is simulated at 10MHz operation frequency, which is practical for asynchronous baseband processing of any signal with bandwidth below 5MHz, including GPS. The CMOS version is constructed by cascading 2-input PCHB AND modules in a binary tree structure whereas the NEM relay version implements the optimized AND function of all inputs

in a single module as shown in Figure 40 and Figure 42. The NEM relays are minimum sized while the average transistor size is 4.5X the minimum length as some transistors have to be strong enough to overcome the staticizers. The simulation result in Figure 43 shows that as the number of inputs increases, the average energy consumed by the CMOS version rises much more rapidly than the NEM relay version, although they start off having comparable energy at 2-bits. In fact, for a 32-bit AND, the NEM relay implementation consumes about 25X lower energy compared to CMOS. The energy savings comes from not only the relays' low switching energy but also logic simplification and compact implementation without bubble-reshuffling or cascading. For the 32-bit PCHB AND, we need almost an order of magnitude fewer relays to build compared to transistors.

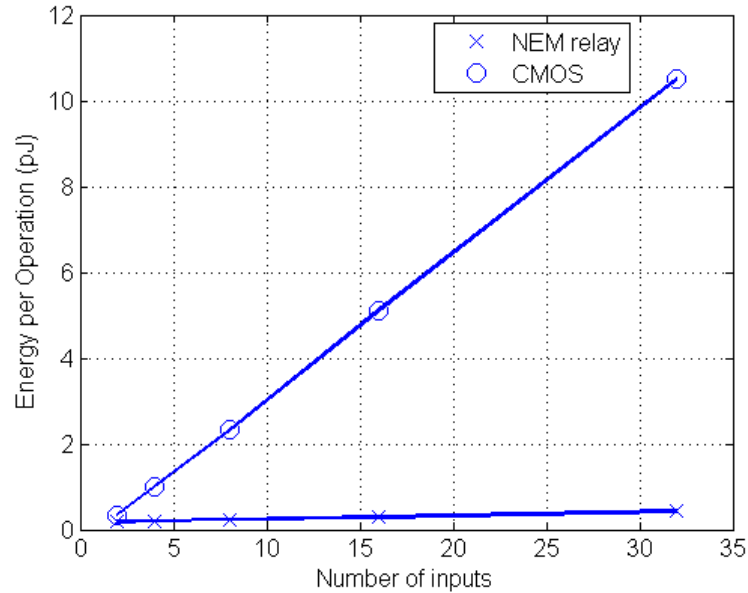


Figure 43 – Average energy per operation at 10MHz for PCHB AND with different number of inputs, implemented with NEM relays and CMOS with 25fF output loads

Studies in [21] and our power simulation results show that NEM relays are

indeed very energy-efficient. The main drawback is the device latency due to mechanical delays. Hence there is an obvious trade-off between energy and latency when comparing NEM relays and CMOS. However, while NEM relay technology can still scale further to improve both its energy and latency, CMOS technology cannot do so without significantly increasing leakage energy. Hence a hybrid CMOS and NEM relays approach can be used to balance the strengths and weaknesses in both technologies. High performance sub-systems can continue to use CMOS but power-gated with NEM relays [28]. Power gating of asynchronous circuits is attractive because of Zero-Delay Ripple Turn On (ZDRTO) [20]. In synchronous circuits, upon system wake-up, processing can only begin after stabilization of the voltage supply so that all system timing requirements are met. In asynchronous systems, ZDRTO kicks start computation just as the pipelines are waking up. This allows asynchronous systems to be powered down easily when idle and woken up efficiently to process incoming data. The combination of NEM relay power gating and ZDRTO in asynchronous circuits is ideal for power-cycling baseband processing modules towards the backend which are typically idle for milliseconds or longer when waiting for correlation data from modules near the frontend.

Even though the latency in NEM relays tends to limit their use below 100MHz, they are fast enough for many baseband processing applications. GPS baseband processing is an example of such an application where NEM relays can completely replace CMOS.

7.3 Advanced Correlators

7.3.1 NEMS Correlators

NEM relays can be used to implement ultra-low-power correlators. NEM relays consume no static power and allow for flexible design implementations that further reduce circuit complexity and thus dynamic power. They are also fast enough for GPS applications.

7.3.2 Weak-signal Integration

To increase the carrier-to-noise ratio, the correlators in the GPS baseband processor need to integrate longer than the minimum period required. Consequently, correlators must be able to handle over 20 bits of correlation result. In a conventional design, a larger correlator would mean higher power consumption but in the case of our optimized counter-based accumulator design, a larger correlator only results in negligible power increase because it translates to just a larger counter for the bits of higher significance that do not switch often.

Our baseband processor is equipped with the data decoding subsystem to extract the navigation message in the received signal and to determine bit-lock and frame-lock. A control signal from this module can inform the correlators if they are integrating across data bit transition boundary. This control signal in turn gates the dump signal of the accumulators.

7.3.3 GNSS Interoperability

The more satellites the receiver sees the more accurate and reliable its navigation solution becomes. The receiver also has more information to implement more robust spoofing-detection and jamming-resistant algorithms. Therefore, to maximize the number of satellites available to the receiver so that it tracks the ones with the best signal integrity, the receiver should not limit its tracking capability to just the GPS L1 C/A civil signal but the entire Global Navigation Satellite System (GNSS). The GNSS comprises the United States GPS constellation, the Russian GLONASS and the up-and-coming European Union's Galileo, and Chinese Beidou and Compass systems. Because these constellations broadcast at different frequency bands, both the GPS baseband processor and the RF frontend design must be flexible enough to accommodate them. Since each subsystem in our asynchronous GPS baseband processor only runs as fast as it needs to run, it is already optimized for any frontend frequency plan needed for inter-constellation operability. Besides, voltage scaling can be applied to different subsystems to accommodate the various maximum operation frequencies of the subsystems while consuming minimal power. For example, according to Figure 25 and Figure 26, the correlators processing the GPS L5 signal (bandwidth 20MHz) may be powered at 1.16V to accommodate sampling frequencies above 40MHz whereas the correlators processing the GPS L1 C/A signal (bandwidth 2MHz) may be powered at 0.84V.

CONCLUSIONS

In this dissertation, I have introduced the concept of asynchrony in the GPS receivers. These intra-channel, inter-channel and computation asynchronies are not conducive to synchronous baseband processing. By using asynchronous VLSI techniques, all subsystems in the baseband processor are optimized to run at their natural frequencies without clocking. The data stream from the frontend *is* the clock and data tokens are passed down from one subsystem to the next by handshakes. I introduced the design and implementation of an asynchronous GPS baseband processor and have shown that it can attain power levels as low as 1.4mW during continuous tracking, with 3D rms position error below 4m. I described micro-architecture level optimizations for exploiting asynchrony and power reduction in baseband processing. They include reducing switching activity down to the bit level in the counter-based accumulators, employing arbitrated tracking loops with deferred updates, and implementing fixed-point arithmetic and mathematical approximations as bundled-data logic blocks. Synchronous designers can emulate and benefit from our approach by breaking the full system down into subsystems based on their data rates and driving them with fast and slow clocks. However, synchronous designs will have more difficulty implementing completely data-driven behavior due to the system clock ratios involved.

In addition, I also introduced advanced technologies for future implementations. Smaller technology nodes can be used to reduce dynamic power consumption in transistors. However, for technologies below 45nm, leakage becomes a concern and power gating techniques need to be applied or a new technology such as

NEM relays be adopted. I demonstrated ways to leverage strengths of NEM relays to further enhance asynchronous circuit design. As asynchronous design helps with dynamic power, NEM relays help with static power. In addition, the flexibility of NEM relay implementations permits further reduction in circuit complexity, device count and thus dynamic power as well. Hence, with our asynchronous system design, together with emerging technologies such as asynchronous NEMS, we can extend our exploitation of asynchrony to enable truly ultra-low power operation in not just GPS signals alone but also the rest of the GNSS signals.

BIBLIOGRAPHY

- [1] J. A. Brzozowski and C.-J. H. Seger, Asynchronous circuits, ser. Monographs in computer science. Springer, 1995.
- [2] B. Sheikh and R. Manohar, “An operand-optimized asynchronous ieee 754 double-precision floating-point adder,” in Asynchronous Circuits and Systems (ASYNC), 2010 IEEE Symposium on, may 2010, pp. 151 –162.
- [3] A. Martin, “Remarks on low-power advantages of asynchronous circuits,” in Proc. European Solid-State Circuits Conference (ESSCIRC), 1998.
- [4] K.-W. Cheng, K. Natarajan, and D. Allstot, “A 7.2mw quadrature gps receiver in 0.13um cmos,” in Solid-State Circuits Conference – Digest of Technical Papers, 2009. ISSCC 2009. IEEE International, feb. 2009, pp. 422 –423,423a.
- [5] B. O’Hanlon, T. Humphreys, M. Psiaki, and P. Kintner, Jr., “Development and field testing of a dsp-based dual-frequency software gps receiver,” in Proc. ION GNSS 2009, 2009, pp. 317–325.
- [6] J.-M. Wei, C.-N. Chen, K.-T. Chen, C.-F. Kuo, B.-H. Ong, C.-H. Lu, C.-C. Liu, H.-C. Chiou, H.-C. Yeh, J.-H. Shieh, K.-S. Huang, K.-I. Li, M.-J. Wu, M.-H. Li, S.-H. Chou, S.-L. Chew, W.-L. Lien, W.-G. Yau, W.-Z. Ge, W.-C. Lai, W.-H. Ting, Y.-J. Tsai, Y.-C. Yen, and Y.-C. Yeh, “A 110nm rfcmos gps soc with 34mw -165dbm tracking sensitivity,” in Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International, feb. 2009, pp. 254 – 255,255a.
- [7] W. Namgoong, S. Reader, and T. Meng, “An all-digital low-power if gps synchronizer,” Solid-State Circuits, IEEE Journal of, vol. 35, no. 6, pp. 856 –864, jun 2000.
- [8] B. Parkinson and J. Spilker, The global positioning system: theory and applications, ser. Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, 1996, no. v. 1; v. 163.

- [9] K. Borre, D. Akos, N. Bertelsen, P. Rinder, and S. Jensen, A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach. Boston, MA: Birkhuser, 2006.
- [10] P. Misra and P. Enge, Global Positioning System: Signals, Measurements, and Performance. Lincoln, MA: Ganga-Jamuna Press, 2006.
- [11] E. Kaplan and C. Hegarty, Understanding GPS: Principles and Applications, Second Edition. Norwood, MA: Artech House, 2005.
- [12] C. L. Seitz, "System timing," in Introduction to VLSI Systems, C. A. Mead and L. A. Conway, Eds. Addison Wesley, 1980, ch. 7.
- [13] B. K. Levitt and G. A. Morris, "An improved digital algorithm for fast amplitude approximations of quadrature pairs," DSN Progress Report 42-40, pp. 98–101, 1977.
- [14] G. Gramegna, P. Mattos, M. Losi, S. Das, M. Franciotta, N. Bellantone, M. Vaiana, V. Mandara, and M. Paparo, "A 56-mw 23-mm² single-chip 180-nm cmos gps receiver with 27.2-mw 4.1-mm² radio," Solid-State Circuits, IEEE Journal of, vol. 41, no. 3, pp. 540 – 551, march 2006.
- [15] F. Van Diggelen, A-GPS: Assisted GPS, GNSS, and SBAS. Artech House, 2009.
- [16] R. Manohar and A. Martin, "Quasi-Delay-Insensitive Circuits are Turing-Complete" in Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1995.
- [17] Market Info Group LLC. <http://www.marketinfogroup.com/location-based-services-market-technology/>
- [18] Karmazin, R.; Otero, C.T.O.; Manohar, R., "cellTK: Automated Layout for Asynchronous Circuits with Nonstandard Cells," Asynchronous Circuits and Systems (ASYNC), 2013 IEEE 19th International Symposium on , vol., no., pp.58,66, 19-22 May 2013.

- [19] LaFrieda, C.; Manohar, R., "Reducing Power Consumption with Relaxed Quasi Delay-Insensitive Circuits," *Asynchronous Circuits and Systems*, 2009. ASYNC '09. 15th IEEE Symposium on , vol., no., pp.217,226, 17-20 May 2009.
- [20] Ortega, C.; Tse, J.; Manohar, R., "Static Power Reduction Techniques for Asynchronous Circuits," *Asynchronous Circuits and Systems (ASYNC)*, 2010 IEEE Symposium on , vol., no., pp.52,61, 3-6 May 2010.
- [21] M. Spencer, F. Chen, C. Wang, R. Nathanael, H. Fariborzi, A. Gupta, H. Kam, V. Pott, J. Jeon, T. Liu, D. Markovic', E. Alon, V.Stojanovic', "Demonstration of Integrated Micro-Electro-Mechanical Relay Circuits for VLSI Applications", *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, January 2011.
- [22] Chen, F.; Hei Kam; Markovic, D.; Tsu-Jae King Liu; Stojanovic, V.; Alon, E., "Integrated circuit design with NEM relays," *Computer-Aided Design*, 2008. ICCAD 2008. IEEE/ACM International Conference on , vol., no., pp.750,757, 10-13 Nov. 2008.
- [23] Sheikh, Basit Riaz and Manohar, Rajit, "Energy-Efficient Pipeline Templates for High-Performance Asynchronous Circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 7, December 2011.
- [24] Sutherland, I.; Fairbanks, S., "GasP: a minimal FIFO control," *Asynchronous Circuits and Systems*, 2001. ASYNC 2001. Seventh International Symposium on, vol., no., pp.46,53, 2001.
- [25] Ferretti, M.; Beerel, P.A., "Single-track asynchronous pipeline templates using 1-of-N encoding," *Design, Automation and Test in Europe Conference and Exhibition*, 2002. Proceedings , vol., no., pp.1008,1015, 2002.
- [26] Amponsah, K.; Yoshimizu, Norimasa; Ardanuc, S.; Lal, A., "Near-kT switching-energy lateral NEMS switch," *Nano/Micro Engineered and Molecular Systems (NEMS)*, 2010 5th IEEE International Conference on , vol., no., pp.985,988, 20-23 Jan. 2010.
- [27] Daesung Lee; Lee, W.S.; Chen Chen; Fallah, F.; Provine, J.; Soogine Chong; Watkins, J.; Howe, R.T.; Wong, H.-S.P.; Mitra, S., "Combinational Logic Design Using Six-Terminal NEM Relays," *Computer-Aided Design of Integrated Circuits and Systems*, *IEEE Transactions on* , vol.32, no.5, pp.653,666, May 2013.

- [28] Fariborzi, H.; Spencer, M.; Karkare, V.; Jaeseok Jeon; Nathanael, R.; Chengcheng Wang; Chen, F.; Hei Kam; Pott, V.; Tsu-Jae King Liu; Alon, E.; Stojanovic, V.; Markovic, D., "Analysis and demonstration of MEM-relay power gating," Custom Integrated Circuits Conference (CICC), 2010 IEEE , vol., no., pp.1,4, 19-22 Sept. 2010.
- [29] Nowick, S.M.; Singh, M., "High-Performance Asynchronous Pipelines: An Overview," Design & Test of Computers, IEEE , vol.28, no.5, pp.8,22, Sept.-Oct. 2011
- [30] A. M. Lines, "Pipelined Asynchronous Circuits", Master Thesis, California Institute of Technology, June 1998.
- [31] Verizon. <https://community.verizonwireless.com/thread/583734/>
- [32] Electronic Design. "What Drains The Smart-Phone Battery?", <http://electronicdesign.com/ed-europe/what-drains-smart-phone-battery>.
- [33] Hardforum. "Why Does GPS Drain the Battery So Much?", <http://hardforum.com/showthread.php?t=1764489>
- [34] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10, (Berkeley,CA, USA), pp. 21, USENIX Association, 2010.
- [35] M. Lombardi, L. Nelson, A. Novick and V. Zhang, "Time and Frequency Measurements Using the Global Positioning System", in The International Journal of Metrology, pp. 26-33, Jul, Aug, Sept, 2001